

## PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-322306

(43)Date of publication of application : 24.11.2000

(51)Int.Cl.

G06F 12/00  
G06F 13/00

(21)Application number : 11-143502

(71)Applicant : FUJITSU LTD

(22)Date of filing : 24.05.1999

(72)Inventor : SHINKAI YOSHITAKE  
TSUCHIYA YOSHIHIRO  
MURAKAMI TAKEO

(30)Priority

Priority number : 10328463  
11063589Priority date : 18.11.1998  
10.03.1999

Priority country : JP

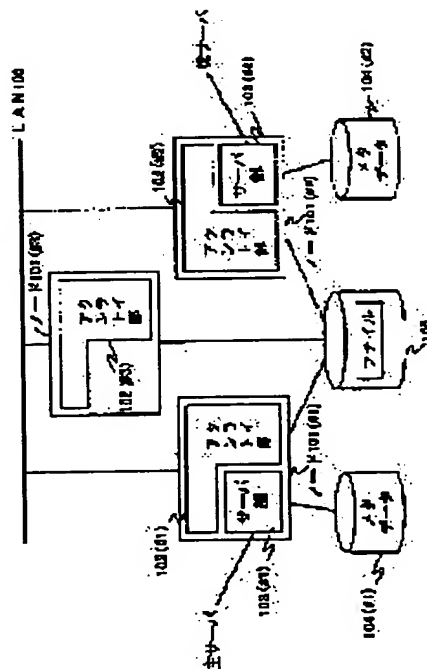
JP

## (54) INTER-NODE SHARED FILE CONTROL SYSTEM

## (57)Abstract:

**PROBLEM TO BE SOLVED:** To reduce the increase of overhead and the deterioration of system performance accompanied by file access, and to prevent the inversion of a file time at the time of server switching in a duplex server system in an inter-node shared file system with tokens.

**SOLUTION:** At the time of requesting a token from a client part 102 to a server 103, the server part 103 responds with the token of a whole file to the client part 102 when any competition between the plural clients 102 is not generated. Only at the time of performing access to the final block of the file, the client part 102 captures the size token corresponding to the file from the server part 103, and performs access to the final block. The server part 103 can simultaneously respond with the time token of write authority to permit the change of the file time to the plural client parts 102. The client part 102 obtains the time token of the write authority, and then executes file access without inquiring for the file time to the server part 103. The server part 103 collects the time token of the write authority from the client part 102 in a prescribed timing, and updates the file time managed by itself.



## LEGAL STATUS

[Date of request for examination] 29.10.2002

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than

the examiner's decision of rejection or  
application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision  
of rejection]

[Date of requesting appeal against examiner's  
decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2000-322306

(P2000-322306A)

(43) 公開日 平成12年11月24日 (2000. 11. 24)

(51) Int.Cl. <sup>7</sup>	識別記号	F I	テーマコード(参考)
G 0 6 F 12/00	5 3 5	G 0 6 F 12/00	5 3 5 C 5 B 0 8 2
	5 3 1		5 3 1 D 5 B 0 8 9
13/00	3 5 1	13/00	3 5 1 E

審査請求 未請求 請求項の数35 O L (全 27 頁)

(21) 出願番号 特願平11-143502

(22) 出願日 平成11年5月24日 (1999. 5. 24)

(31) 優先権主張番号 特願平10-328463

(32) 優先日 平成10年11月18日 (1998. 11. 18)

(33) 優先権主張国 日本 (J P)

(31) 優先権主張番号 特願平11-63589

(32) 優先日 平成11年3月10日 (1999. 3. 10)

(33) 優先権主張国 日本 (J P)

(71) 出願人 000005223

富士通株式会社

神奈川県川崎市中原区上小田中4丁目1番1号

(72) 発明者 新関 慶武

神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

(72) 発明者 土屋 芳浩

神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

(74) 代理人 100074099

弁理士 大菅 義之 (外1名)

最終頁に続く

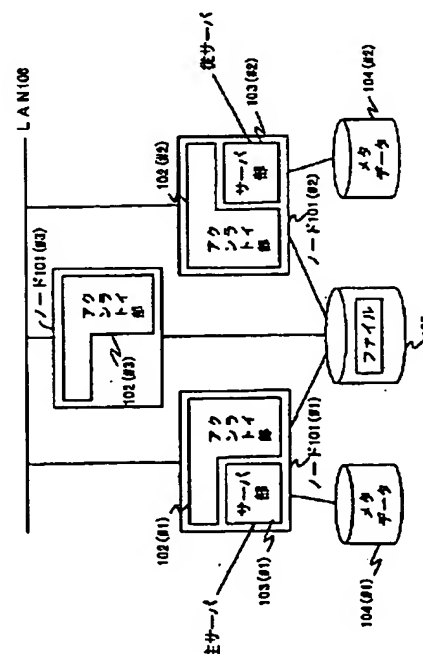
(54) 【発明の名称】 ノード間共用ファイル制御方式

(57) 【要約】

【課題】 トークンによるノード間共用ファイルシステムにおいて、ファイルアクセスに伴うオーバーヘッドの増大とシステム性能の低下を抑制し、二重化サーバシステムでのサーバ切替え時のファイル時刻の逆転を防止することにある。

【解決手段】 クライアント部102からサーバ部103へのトークンの要求時に、103は、複数の102間で競合が無ければ、103から102へファイル全体のトークンを応答する。102は、ファイルの最終ブロックへのアクセス時のみ、103からそのファイルに対応するサイズトークンを獲得した上でその最終ブロックにアクセスする。103は、ファイル時刻の変更を許容するwrite 権の時刻トークンを複数の102に同時に応答できる。102は、write 権の時刻トークンを獲得した後は、103にファイル時刻を問い合わせることなく、ファイルアクセスを実行する。103は、所定のタイミングで、102からwrite 権の時刻トークンを回収し、自身が管理するファイル時刻を更新する。

本発明の実施の形態のシステム構成図



**【特許請求の範囲】**

**【請求項 1】** ユーザプログラムからのファイル操作要求を受けて、1つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイル制御方法であって、前記クライアント装置から前記サーバ装置へのトークンの要求時に、前記サーバ装置において複数の前記クライアント装置間での該トークンの競合の有無を判定し、該競合が無ければ、前記サーバ装置から前記クライアント装置へファイル全体のトークンを応答する、過程を含むことを特徴とするノード間共用ファイル制御方法。

**【請求項 2】** ユーザプログラムからのファイル操作要求を受けて、1つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該サーバ装置であって、前記クライアント装置から前記サーバ装置へのトークンの要求時に、複数の前記クライアント装置間での該トークンの競合の有無を判定する判定手段と、該判定手段が該競合が無いと判定した場合に、前記クライアント装置へファイル全体のトークンを応答する応答手段と、を含むことを特徴とするサーバ装置。

**【請求項 3】** ユーザプログラムからのファイル操作要求を受けて、1つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該サーバ装置であるコンピュータにより使用されたときにそれによって読み出されるプログラムを記録した記録媒体であって、前記クライアント装置から前記サーバ装置へのトークンの要求時に、複数の前記クライアント装置間での該トークンの競合の有無を判定する機能と、該判定手段が該競合が無いと判定した場合に、前記クライアント装置へファイル全体のトークンを応答する機能と、を前記コンピュータに行わせるためのプログラムを記録したコンピュータ読み出し可能記録媒体。

**【請求項 4】** ユーザプログラムからのファイル操作要求を受けて、1つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とす

るノード間共用ファイル制御方法であって、前記クライアント装置と前記サーバ装置の間で、ファイル時刻を制御するための時刻トークンを通信し、前記サーバ装置において、前記ファイル時刻の変更を許容するwrite 権の時刻トークンを複数の前記クライアント装置に同時に応答する制御を実行し、前記クライアント装置において、前記write 権の時刻トークンを獲得した後は、前記サーバ装置にファイル時刻を問い合わせることなく、ファイルアクセスを実行し、前記サーバ装置において、所定のタイミングで前記クライアント装置から前記write 権の時刻トークンを回収し、自身が管理するファイル時刻を更新する、過程を含むことを特徴とするノード間共用ファイル制御方法。

**【請求項 5】** ユーザプログラムからのファイル操作要求を受けて、1つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該クライアント装置であって、前記サーバ装置との間で、ファイル時刻を制御するための時刻トークンを通信する通信手段と、前記サーバ装置から複数の前記クライアント装置に同時に応答され得るトークンであって前記ファイル時刻の変更を許容するwrite 権の時刻トークンを前記サーバ装置から獲得した後は、前記サーバ装置にファイル時刻を問い合わせることなく、ファイルアクセスを実行するアクセス制御手段と、を含むことを特徴とするクライアント装置。

**【請求項 6】** ユーザプログラムからのファイル操作要求を受けて、1つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該クライアント装置であるコンピュータにより使用されたときにそれによって読み出されるプログラムを記録した記録媒体であって、前記サーバ装置との間で、ファイル時刻を制御するための時刻トークンを通信する機能と、前記サーバ装置から複数の前記クライアント装置に同時に応答され得るトークンであって前記ファイル時刻の変更を許容するwrite 権の時刻トークンを前記サーバ装置から獲得した後は、前記サーバ装置にファイル時刻を問い合わせることなく、ファイルアクセスを実行する機能と、を前記コンピュータに行わせるためのプログラムを記録したコンピュータ読み出し可能記録媒体。

**【請求項 7】** ユーザプログラムからのファイル操作要

求を受けて、1つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該サーバ装置であって、

前記クライアント装置との間で、ファイル時刻を制御するための時刻トークンを通信する通信手段と、

前記ファイル時刻の変更を許容するwrite 権の時刻トークンを複数の前記クライアント装置に同時に応答する制御を実行する応答手段と、

所定のタイミングで前記クライアント装置から前記write 権の時刻トークンを回収し、自身が管理するファイル時刻を更新するファイル時刻更新手段と、

を含むことを特徴とするサーバ装置。

【請求項8】 ユーザプログラムからのファイル操作要求を受けて、1つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該サーバ装置であるコンピュータにより使用されたときにそれによって読み出されるプログラムを記録した記録媒体であって、

前記クライアント装置との間で、ファイル時刻を制御するための時刻トークンを通信する機能と、

前記ファイル時刻の変更を許容するwrite 権の時刻トークンを複数の前記クライアント装置に同時に応答する制御を実行する機能と、

所定のタイミングで前記クライアント装置から前記write 権の時刻トークンを回収し、自身が管理するファイル時刻を更新する機能と、

を前記コンピュータに行わせるためのプログラムを記録したコンピュータ読み出し可能記録媒体。

【請求項9】 ユーザプログラムからのファイル操作要求を受けて、1つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイル制御方法であって、

前記クライアント装置と前記サーバ装置の間で、ファイルサイズの拡張を制御するためのサイズトークンを通信し、

前記クライアント装置において、ファイルの最終ブロックにアクセスする場合においてのみ、前記サーバ装置から該ファイルに対応するサイズトークンを獲得した上で該最終ブロックにアクセスする、

過程を含むことを特徴とするノード間共用ファイル制御方法。

【請求項10】 ユーザプログラムからのファイル操作

要求を受けて、1つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該クライアント装置であって、

前記サーバ装置との間で、ファイルサイズの拡張を制御するためのサイズトークンを通信する通信手段と、

ファイルの最終ブロックにアクセスする場合においてのみ、前記サーバ装置から該ファイルに対応するサイズトークンを獲得した上で該最終ブロックにアクセスするアクセス手段と、

を含むことを特徴とするクライアント装置。

【請求項11】 ユーザプログラムからのファイル操作要求を受けて、1つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該クライアント装置であるコンピュータにより使用されたときにそれによって読み出されるプログラムを記録した記録媒体であって、

前記サーバ装置との間で、ファイルサイズの拡張を制御するためのサイズトークンを通信する機能と、

ファイルの最終ブロックにアクセスする場合においてのみ、前記サーバ装置から該ファイルに対応するサイズトークンを獲得した上で該最終ブロックにアクセスする機能と、

を前記コンピュータに行わせるためのプログラムを記録したコンピュータ読み出し可能記録媒体。

【請求項12】 ユーザプログラムからのファイル操作要求を受けて、1つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイル制御方法であって、

前記クライアント装置と前記サーバ装置の間で、ファイルデータのアクセスを制御するためのデータトークンを通信し、

該データトークンの通信時に、該データトークンに対応するファイルのディスク上での位置を示すエクステンション情報を通信する、

過程を含むことを特徴とするノード間共用ファイル制御方法。

【請求項13】 ユーザプログラムからのファイル操作要求を受けて、1つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該クライ

アント装置であって、

前記サーバ装置との間で、ファイルデータのアクセスを制御するためのデータトークンを通信する第1の通信手段と、

該データトークンの通信時に、該データトークンに対応するファイルのディスク上での位置を示すエクステント情報を通信する第2の通信手段と、

を含むことを特徴とするクライアント装置。

【請求項14】 ユーザプログラムからのファイル操作要求を受けて、1つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該クライアント装置であるコンピュータにより使用されたときにそれによって読み出されるプログラムを記録した記録媒体であって、

前記サーバ装置との間で、ファイルデータのアクセスを制御するためのデータトークンを通信する機能と、

該データトークンの通信時に、該データトークンに対応するファイルのディスク上での位置を示すエクステント情報を通信する機能と、

を前記コンピュータに行わせるためのプログラムを記録したコンピュータ読み出し可能記録媒体。

【請求項15】 ユーザプログラムからのファイル操作要求を受けて、1つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該サーバ装置であって、

前記クライアント装置との間で、ファイルデータのアクセスを制御するためのデータトークンを通信する第1の通信手段と、

該データトークンの通信時に、該データトークンに対応するファイルのディスク上での位置を示すエクステント情報を通信する第2の通信手段と、

を含むことを特徴とするサーバ装置。

【請求項16】 ユーザプログラムからのファイル操作要求を受けて、1つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該サーバ装置であるコンピュータにより使用されたときにそれによって読み出されるプログラムを記録した記録媒体であって、

前記クライアント装置との間で、ファイルデータのアクセスを制御するためのデータトークンを通信する機能と、

該データトークンの通信時に、該データトークンに対応するファイルのディスク上での位置を示すエクステント情報を通信する機能と、

を前記コンピュータに行わせるためのプログラムを記録したコンピュータ読み出し可能記録媒体。

【請求項17】 請求項1、2、4、5、7、9、10、12、13、又は15のいずれか1項に記載のノード間共用ファイルシステムにおいて前記サーバ装置が二重化される構成を有するノード間共用ファイル制御方法であって、

主系のサーバ装置においてファイル時刻が設定される際に、該ファイル時刻を従系のサーバ装置に送信し、

該従系のサーバ装置において、該ファイル時刻を設定する、

過程を含むことを特徴とするノード間共用ファイル制御方法。

【請求項18】 ユーザプログラムからのファイル操作要求を受けて、1つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイル制御方法であって、

前記サーバ装置において、複数のノードから共用される1つ以上のディスクボリューム毎に、空きディスク領域群、使用中ディスク領域群、及び前記各クライアント装置に対応するリザーブ中ディスク領域群を管理し、

前記クライアント装置において、前記サーバ装置に対して、ディスク領域のリザーブを要求し、

該リザーブ要求に対して、前記サーバ装置において、前記空きディスク領域群からディスク領域をリザーブ中ディスク領域として確保し、それに関する情報を該リザーブ要求を発行したクライアント装置に通知すると共に、該確保したリザーブ中ディスク領域を前記リザーブ要求を発行したクライアント装置に対応する前記リザーブ中ディスク領域群として管理し、

前記リザーブ要求を発行したクライアント装置において、該リザーブ要求に回答して前記サーバ装置から通知された情報に対応するリザーブ中ディスク領域をリザーブ中ディスク領域群として管理し、

前記クライアント装置において、ユーザプログラムによるファイルへのデータ書き出し要求に伴って新たなディスク領域を割り当てる必要が発生した場合に、該クライアント装置が管理する前記リザーブ中ディスク領域群から最適なリザーブ中ディスク領域を選択し、そこに対してデータ書き出しを実行し、該リザーブ中ディスク領域を前記リザーブ中ディスク領域群としての管理からはずし、該データ書き出しを実行したリザーブ中ディスク領域に関する情報を前記サーバ装置に通知し、

前記サーバ装置において、前記クライアント装置から通知された情報に対応する前記データ書き出しが発生したり

ザーブ中ディスク領域を、該通知を行ったクライアント装置に対応する前記リザーブ中ディスク領域群としての管理からはずして前記使用中ディスク領域として管理する、  
過程を含むことを特徴とするノード間共用ファイル制御方法。

【請求項19】 請求項18に記載の方法であって、前記サーバ装置における前記空きディスク領域群及びリザーブ中ディスク領域群の管理と、前記クライアント装置における前記リザーブ中ディスク領域群の管理を、前記ディスク領域の複数のサイズ範囲毎に行う、過程を更に含むことを特徴とするノード間共用ファイル制御方法。

【請求項20】 請求項18に記載の方法であって、前記クライアント装置において、前記ユーザプログラムによるファイルへのデータ書出し要求に伴って新たなディスク領域を割り当てる必要が発生した場合に、該クライアント装置が管理する前記リザーブ中ディスク領域群から前記ファイルへのデータ書出しが既に行われているディスク領域に連続するリザーブ中ディスク領域を選択し、該選択に失敗した場合には、前記サーバ装置に対して、該連続するリザーブ中ディスク領域のリザーブ要求を発行する、  
過程を更に含むことを特徴とするノード間共用ファイル制御方法。

【請求項21】 請求項18に記載の方法であって、前記サーバ装置において、前記クライアント装置の障害を監視し、その結果障害が検出されたクライアント装置に対応する前記リザーブ中ディスク領域群を、全て前記空きディスク領域群に変更する、  
過程を更に含むことを特徴とするノード間共用ファイル制御方法。

【請求項22】 請求項18に記載の方法であって、前記クライアント装置において、それが管理する前記リザーブ中ディスク領域群中のディスク領域が所定量を下回ったときに、前記サーバ装置に対して、新たなディスク領域のリザーブ要求を発行する、  
過程を更に含むことを特徴とするノード間共用ファイル制御方法。

【請求項23】 請求項18に記載の方法であって、前記クライアント装置において、前記ユーザプログラムによるファイルへのデータ書出し要求に基づいて書き出されるデータを、主記憶上にキャッシュし、前記リザーブ中ディスク領域の割り当てを遅延させる、  
過程を更に含むことを特徴とするノード間共用ファイル制御方法。

【請求項24】 請求項18に記載の方法であって、前記クライアント装置において、前記データ書出しを実行したリザーブ中ディスク領域に関する情報の前記サーバ装置への通知は、前記ユーザプログラムがファイルを

クローズし、又はキャッシュが一杯になり、或いは前記サーバ装置からデータトークンの回収を要求されるタイミングで行う、

過程を更に含むことを特徴とするノード間共用ファイル制御方法。

【請求項25】 ユーザプログラムからのファイル操作要求を受けて、1つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該クライアント装置であって、

前記サーバ装置に対して、ディスク領域のリザーブを要求するリザーブ要求手段と、

該リザーブ要求に応答して前記サーバ装置から通知された情報に対応するリザーブ中ディスク領域をリザーブ中ディスク領域群として管理するリザーブ中ディスク領域群管理手段と、

ユーザプログラムによるファイルへのデータ書出し要求に伴って新たなディスク領域を割り当てる必要が発生した場合に、前記リザーブ中ディスク領域群管理手段が管理する前記リザーブ中ディスク領域群から最適なりザーブ中ディスク領域を選択し、そこに対してデータ書出しを実行し、前記リザーブ中ディスク領域群管理手段に対して該データ書出しを実行したリザーブ中ディスク領域の管理をはずさせ、該リザーブ中ディスク領域に関する情報を前記サーバ装置に通知するクライアント側データ書出し制御手段と、

を含むことを特徴とするクライアント装置。

【請求項26】 ユーザプログラムからのファイル操作要求を受けて、1つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該クライアント装置であるコンピュータにより使用されたときにそれによって読み出されるプログラムを記録した記録媒体であって、

前記サーバ装置に対して、ディスク領域のリザーブを要求する機能と、

該リザーブ要求に応答して前記サーバ装置から通知された情報に対応するリザーブ中ディスク領域をリザーブ中ディスク領域群として管理する機能と、

ユーザプログラムによるファイルへのデータ書出し要求に伴って新たなディスク領域を割り当てる必要が発生した場合に、自身が管理する前記リザーブ中ディスク領域群から最適なりザーブ中ディスク領域を選択し、そこに対してデータ書出しを実行し、該リザーブ中ディスク領域を前記リザーブ中ディスク領域群としての管理からはずし、該データ書出しを実行したリザーブ中ディスク領域



域に関する情報を前記サーバ装置に通知する機能と、  
を前記コンピュータに行わせるためのプログラムを記録したコンピュータ読出し可能記録媒体。

【請求項 27】 ユーザプログラムからのファイル操作要求を受けて、1つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成するサーバ装置であって、

複数のノードから共用される1つ以上のディスクボリューム毎に、空きディスク領域群、使用中ディスク領域群、及び前記各クライアント装置に対応するリザーブ中ディスク領域群を管理するディスク領域管理手段と、  
前記クライアント装置からのディスク領域のリザーブ要求に対して、前記空きディスク領域群からディスク領域をリザーブ中ディスク領域として確保し、それに関する情報を該リザーブ要求を発行したクライアント装置に通知すると共に、該確保したリザーブ中ディスク領域を前記リザーブ要求を発行したクライアント装置に対応する前記リザーブ中ディスク領域群として管理するリザーブ中ディスク領域確保手段と、

前記クライアント装置から通知された情報に対応するデータ書出しが発生したリザーブ中ディスク領域を、該通知を行ったクライアント装置に対応する前記リザーブ中ディスク領域群としての管理からはずして前記使用中ディスク領域として管理するサーバ側データ書出し制御手段と、

を含むことを特徴とするサーバ装置。

【請求項 28】 ユーザプログラムからのファイル操作要求を受けて、1つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成するサーバ装置であるコンピュータにより使用されたときにそれによって読み出されるプログラムを記録した記録媒体であって、

複数のノードから共用される1つ以上のディスクボリューム毎に、空きディスク領域群、使用中ディスク領域群、及び前記各クライアント装置に対応するリザーブ中ディスク領域群を管理する機能と、  
前記クライアント装置からのディスク領域のリザーブ要求に対して、前記空きディスク領域群からディスク領域をリザーブ中ディスク領域として確保し、それに関する情報を該リザーブ要求を発行したクライアント装置に通知すると共に、該確保したリザーブ中ディスク領域を前記リザーブ要求を発行したクライアント装置に対応する前記リザーブ中ディスク領域群として管理する機能と、  
前記クライアント装置から通知された情報に対応するデ

ータ書出しが発生したリザーブ中ディスク領域を、該通知を行ったクライアント装置に対応する前記リザーブ中ディスク領域群としての管理からはずして前記使用中ディスク領域として管理する機能と、

を前記コンピュータに行わせるためのプログラムを記録したコンピュータ読出し可能記録媒体。

【請求項 29】 ユーザプログラムからのファイル操作要求を受けて、1つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイル制御方法であって、

前記サーバ装置において、前記クライアント装置からのトークン回収完了メッセージの受信時に、該メッセージに対応するトークン回収の契機となった要求を処理している実行単位が保持していたファイルロックを継承して処理を実行することによりデッドロックを回避する過程を含む、

ことを特徴とするノード間共用ファイル制御方法。

【請求項 30】 請求項 29 に記載の方法であって、前記ロックの継承を行える実行単位を1つに制限する過程を更に含む、

ことを特徴とするノード間共用ファイル制御方法。

【請求項 31】 請求項 29 に記載の方法であって、前記トークン回収の待ち状態を資源として記憶し、他の資源の獲得待ち状態との関係から、デッドロック状態を自動的に検出する過程を更に含む、

ことを特徴とするノード間共用ファイル制御方法。

【請求項 32】 請求項 31 に記載の方法であって、前記デッドロック状態が検出され該状態の原因となっているトランザクションがキャンセルさせられる際に、更新されたキャッシュデータの無効化と共に、主記憶装置に常駐されている関連制御表の再設定を行う過程を更に含む、

ことを特徴とするノード間共用ファイル制御方法。

【請求項 33】 請求項 29 に記載の方法であって、デッドロック状態の発生に備え、ファイル又はディスクに関する属性情報を保持するメタデータの更新をキャッシュ上でのみ行い、ディスクへの書き込みが、要求された処理の完了まで遅延させられるトランザクション制御において、

キャッシュデータの更新時に更新されたキャッシュ位置を記録する過程と、

トランザクションの完了時に、前記記録から必要最小限の変更データのみをログファイルに書き出すことによりログデータ量を削減する過程と、

を含むことを特徴とするノード間共用ファイル制御方法。

【請求項 34】 請求項 33 に記載の方法であって、更新されたキャッシュ位置を記録する際に、該記録と先



行する記録とをマージすることにより、ログファイルに書き出すログデータ量を最小化する過程を更に含む、ことを特徴とするノード間共用ファイル制御方法。

【請求項35】 請求項33に記載の方法であって、前記キャッシュは2次キャッシュを含む、ことを特徴とするノード間共用ファイル制御方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、複数のノード（ホストコンピュータ）から同一のファイルを共用することを可能とするノード間共用ファイルシステム（分散ファイルシステム）のコンシステンシ保証制御技術に関する。

【0002】

【従来の技術】分散ファイルシステムにおいて、トークンを利用して複数のノード上にキャッシュされているデータのコンシステンシ（一貫性、整合性）を保つ方式は良く知られている。代表的な方式では、ファイルのアクセス範囲（通常、ブロック番号の始端と終端が用いられる）ごとにmultiple-read/single-writeの制御を行うトークンが用意される。そして、ファイルにアクセスしようとするノードは、自身がアクセス範囲のトークンを保持しているか否かを調べ、もし保持していなければトークンを管理しているサーバにトークンを要求する。トークンを管理しているサーバは、read権は複数のノードに渡されることを許し（multiple-read）、write権は1つのノードのみに渡されるように（single-write）、アクセス権制御を実行する。

【0003】

【発明が解決しようとする課題】上述の従来方式は、各ノードにキャッシュされているデータの一貫性を保ちつつサーバとクライアントの間の通信を減らすために有効な方式であるが、以下の問題点を有する。

- 1) ファイルアクセスの都度にトークンを獲得する必要がある。例えば、科学技術計算のための巨大なファイルをユーザがシーケンシャルにアクセスする場合、ユーザは、特定バイトずつのファイルアクセス要求を出す都度に、サーバにトークンを獲得するための要求を発行せざるを得ない。この事実は、オーバヘッドの増大を招く。
- 2) ファイルが最後にアクセスされた時刻を保持するファイルアクセス時刻（ファイル時刻）の正当性を保証するために、ユーザはファイルアクセス要求を発行する都度にサーバにそのアクセスの存在を通知せざるを得ない。この事実は、オーバヘッドの増大を招く。
- 3) ユーザはファイルサイズを更新するときにはその旨をサーバに通知し、サーバは他ノードに発行されている全てのトークンを回収しなければならない。このため、例えばファイルを拡張するプログラムとファイルをその先頭から順に読むプログラムをそれぞれ異なるノードで同時に実行させることができず、システム全体の性能が

低下するといった問題が生ずる。

4) サーバが二重化され、障害発生時に運用サーバが待機系サーバに切り替えられる機能を有するシステムにおいて、待機系サーバへの切替えの時点でいまままで運用されてきた時計も待機系のサーバ内の時計に切り替えられるため、ファイル時刻の逆転現象が発生する可能性がある。この事実は、データのコンシステンシの喪失を招く。

5) メインフレームで採用されるような、ディスクがノード間で直接共用されネットワークを介したデータ転送が削減される方式を、離散ファイルの特徴とするオープン系のファイルシステムに適用しようとした場合に、各ノードはファイルシステム上でブロックを割り当てる都度にサーバと通信する必要が生ずる。この事実は、オーバヘッドの増大を招く。

一方、トークンを利用した分散ファイルシステムにおいては、複数のノードが同時並行的なアクセスを行うため、ファイルシステムの耐故障性に関しても十分な配慮が必要である。一般に、ファイルシステムの耐故障性を向上させる方式として、ログファイルを設けてメタデータの更新をトランザクショナルに行うログ方式が知られている。ログ方式では一般に、1つのトランザクションの処理途中結果を他のトランザクションに見せてはならないという制約のために、いわゆる2フェーズロック制御が行われる。この制御では、更新に必要なロックが順に獲得されてゆき、全ての更新が完了した時点で一括して、メタデータの更新内容がログファイルロックに書き出され、書き出しが完了した時点でロックが一括して返却される。この際に必然的に発生する複数のロック獲得に伴うデッドロックは、資源獲得を示す有向グラフを用いて自動的に検出され、デッドロックの原因となっている一方のトランザクションがキャンセルされ、再試行されることにより解消される方式が、一般的に用いられる。

【0004】しかし、上述のようなログ方式をトークンシステムに適用してデッドロックを自動的に検出し回復を図る汎用的な方式は考え出されていない。また、従来のログ方式では、ログがキャッシュブロック単位に採取されると共に、トランザクション終了時にファイルシステムの実更新が発生するため、I/O量が相対的に多くなるという欠陥があった。

【0005】また上記ログ方式では、トランザクションのキャンセル時のデータ復元処理がメタデータのみに限られ、性能向上のために用意されたメモリに常駐する制御表は対象外であるため、プログラム作成が難しいという欠陥も持っていた。

【0006】本発明の課題は、トークンを用いたノード間共用ファイルシステムにおいて、上述の各問題点を解決することにより、ファイルアクセスに伴うオーバヘッドの増大とシステム性能の低下を抑制すると共に、二重

化サーバシステムにおけるサーバ切替え時のファイル時刻の逆転を防止し、更にメタデータの更新をコンシステントにかつデッドロックフリーで行なうことにより従来のログ方式の性能上及びプログラム作成上の問題点を解決することにある。

#### 【0007】

【課題を解決するための手段】本発明は、ユーザプログラムからのファイル操作要求を受けて、1つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上でそのファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイル制御システムを前提とする。

【0008】本発明の第1の態様は、以下の構成を有する。まず、クライアント装置からサーバ装置へのトークンの要求時に、サーバ装置において複数のクライアント装置間でのそのトークンの競合の有無が判定される。

【0009】そして、その競合が無ければ、サーバ装置からクライアント装置へファイル全体のトークンが応答される。以上の構成を有する本発明の第1の態様の構成では、例えばopen要求時等においてファイル全体のトークンが引き渡されることにより、可能な限り新たなトークン要求を行わずにファイルへの連続アクセスが可能となる。データベースアクセス等を除く一般的なファイルアクセスでは、1つのノードからのwrite 要求の発行時に他のノードからread命令が発行される確率は小さい。従って、1つのノードに引き渡されたファイル全体のトークンが回収される確率も低く、ファイルへの連続アクセス時にアクセス単位ごとにトークン要求が不要になることによる性能向上が期待できる。

【0010】本発明の第2の態様は、以下の構成を有する。まず、クライアント装置とサーバ装置の間で、ファイル時刻を制御するための時刻トークンが通信される。

【0011】また、サーバ装置において、ファイル時刻の変更を許容するwrite 権の時刻トークンを複数のクライアント装置に同時に応答する制御が実行される。また、クライアント装置において、write 権の時刻トークンが獲得された後は、サーバ装置にファイル時刻を問い合わせることなく、ファイルアクセスが実行される。

【0012】また、サーバ装置において、所定のタイミングでクライアント装置からwrite権の時刻トークンが回収され、自身が管理するファイル時刻が更新される。以上の構成を有する本発明の第2の態様の構成では、クライアント装置は、ユーザプログラムが1つのファイルに連続アクセスするような場合において、そのファイルへの最終的なアクセスが終了するまでwrite 権の時刻トークンを返却する必要もまたアクセスの有無をサーバ部に通知する必要もなく、他のノードとの間でそのファイルのファイル時刻の同期をとる必要がなくなる。このため、システム全体の性能を向上させることが可能とな

る。

【0013】本発明の第3の態様は、以下の構成を有する。まず、クライアント装置とサーバ装置の間で、ファイルサイズの拡張を制御するためのサイズトークンが通信される。

【0014】そして、クライアント装置において、ファイルの最終ブロックにアクセスする場合においてのみ、サーバ装置からそのファイルに対応するサイズトークンが獲得された上でその最終ブロックにアクセスされる。

【0015】以上の構成を有する本発明の第3の態様の構成では、ファイルの最終ブロックにアクセスするのでなければ、サイズトークンを獲得することなくファイルにアクセスすることが可能となり、これと並行して、他のノードは、サイズトークンを獲得してファイルの最終ブロックにアクセスし、ファイルのサイズを拡張するwrite 操作処理を実行することができる。このため、例えばファイルを拡張するプログラムとファイルをその先頭から順に読むプログラムをそれぞれ異なるノードで同時に実行させることが可能となり、システム全体の性能を向上させることができる。

【0016】本発明の第4の態様は、以下の構成を有する。まず、クライアント装置とサーバ装置の間で、ファイルデータのアクセスを制御するためのデータトークンが通信される。

【0017】そして、そのデータトークンの通信時に、そのデータトークンに対応するファイルのディスク上での位置を示すエクステント情報が通信される。以上の構成を有する本発明の第4の態様の構成では、複数のノードは、ディスク装置内のファイルに、LAN経由ではなく直結された制御・データ線を介してアクセスすることが可能となる。

【0018】本発明の第5の態様は、上述した本発明の第1乃至第4のいずれかの態様の構成を前提として、さらにサーバ装置が二重化される構成を前提とし、以下の構成を有する。

【0019】まず、主系のサーバ装置においてファイル時刻が設定される際に、そのファイル時刻が従系のサーバ装置に送信される。そして、その従系のサーバ装置において、そのファイル時刻が設定される。

【0020】以上の構成を有する本発明の第5の態様の構成では、サーバ切替え時にも、矛盾のないファイル時刻の付与が可能となる。本発明の第6の態様は、以下の構成を有する。

【0021】まず、サーバ装置において、複数のノードから共用される1つ以上のディスクボリューム毎に、空きディスク領域群、使用中ディスク領域群、及び各クライアント装置に対応するリザーブ中ディスク領域群が管理される。このとき、空きディスク領域群の管理が、ディスク領域の複数のサイズ範囲毎に行われるように構成することができる。

【0022】次に、クライアント装置において、サーバ装置に対して、ディスク領域のリザーブが要求される。このとき、クライアント装置において、それが管理するリザーブ中ディスク領域群中のディスク領域が所定量を下回ったときに、サーバ装置に対して、新たなディスク領域のリザーブ要求が発行されるように構成することができる。

【0023】次に、そのリザーブ要求に対して、サーバ装置において、空きディスク領域群からディスク領域がリザーブ中ディスク領域として確保され、それに関する情報がそのリザーブ要求を発行したクライアント装置に通知されると共に、その確保されたリザーブ中ディスク領域がリザーブ要求を発行したクライアント装置に対応するリザーブ中ディスク領域群として管理される。

【0024】続いて、リザーブ要求を発行したクライアント装置において、そのリザーブ要求に応答してサーバ装置から通知された情報に対応するリザーブ中ディスク領域がリザーブ中ディスク領域群として管理される。このとき、クライアント装置において、リザーブ中ディスク領域群の管理が、ディスク領域の複数のサイズ範囲毎に行われるように構成することができる。

【0025】更に、クライアント装置において、ユーザプログラムによるファイルへのデータ書出し要求に伴って新たなディスク領域を割り当てる必要が発生した場合に、そのクライアント装置が管理するリザーブ中ディスク領域群から最適なリザーブ中ディスク領域が選択され、そこに対してデータ書出しが実行され、そのリザーブ中ディスク領域がリザーブ中ディスク領域群としての管理からはずされ、そのデータ書出しを実行したリザーブ中ディスク領域に関する情報がサーバ装置に通知される。この通知は、ユーザプログラムがファイルをクローズし、又はキャッシュが一杯になり、或いはサーバ装置からデータトークンの回収を要求されるタイミングで行うように構成することができる。このとき、ユーザプログラムによるファイルへのデータ書出し要求に基づいて書き出されるデータが、主記憶上にキャッシュされ、リザーブ中ディスク領域の割り当てが遅延させられるように構成することができる。

【0026】そして、サーバ装置において、クライアント装置から通知された情報に対応するデータ書出しが発生したリザーブ中ディスク領域が、その通知を行ったクライアント装置に対応するリザーブ中ディスク領域群としての管理からはずされて使用中ディスク領域として管理される。

【0027】上述の本発明の第6の態様の構成において、クライアント装置において、ユーザプログラムによるファイルへのデータ書出し要求に伴って新たなディスク領域を割り当てる必要が発生した場合に、そのクライアント装置が管理するリザーブ中ディスク領域群からファイルへのデータ書出しが既に行われているディスク領

域に連続するリザーブ中ディスク領域が選択され、その選択に失敗した場合には、サーバ装置に対して、その連続するリザーブ中ディスク領域のリザーブ要求が発行されるように構成することができる。

【0028】また、サーバ装置において、クライアント装置の障害が監視され、その結果障害が検出されたクライアント装置に対応するリザーブ中ディスク領域群が、全て空きディスク領域群に変更されるように構成することができる。

【0029】以上の構成を有する本発明の第6の態様の構成では、クライアント装置は、サーバ装置に問い合わせることなく、新たなディスク領域をファイルに割り当てることが可能となる。このため、クライアント装置とサーバ装置との間の通信回数を削減でき、システム全体の性能を向上させることが可能となる。また、新たに割り当てられたディスク領域は、データが書き込まれた後のクライアント装置からサーバ装置への応答によって初めて、そのファイルのメタデータ等として記憶される。このため、悪意をもってデータを覗くことを防止することが可能となる。

【0030】本発明の第7の構成は、サーバ装置において、クライアント装置からのトークン回収完了メッセージの受信時に、そのメッセージに対応するトークン回収の契機となった要求を処理している実行単位が保持していたファイルロックを継承して処理を実行することによりデッドロックを回避する過程を含むように構成される。この場合に、ロックの継承を行える実行単位を1つに制限する過程を更に含むように構成することができる。

【0031】上述した本発明の第7の構成によれば、トークン制御において、デッドロックの発生を回避することのできる効率的なファイルロック制御が実現される。本発明の第8の構成は、本発明の第7の構成において、トークン回収の待ち状態を資源として記憶し、他の資源の獲得待ち状態との関係から、デッドロック状態を自動的に検出する過程を更に含むように構成される。

【0032】上述した本発明の第8の構成によれば、トークンに基づいてトランザクション制御されているメタデータ等の更新処理におけるデッドロックの発生を適切に検出することができる。

【0033】本発明の第9の構成は、本発明の第8の構成において、デッドロック状態が検出されその状態の原因となっているトランザクションがキャンセルさせられる際に、更新されたキャッシュデータの無効化と共に、主記憶装置に常駐されている関連制御表の再設定を行う過程を更に含むように構成される。

【0034】上述した本発明の第9の構成によれば、トランザクションのキャンセルに伴う常駐制御表の高速なリストアが実現される。本発明の第10の構成は、本発明の第7の構成を前提として、デッドロック状態の発生

に備え、ファイル又はディスクに関する属性情報を保持するメタデータの更新をキャッシュ上でのみ行い、ディスクへの書き込みが、要求された処理の完了まで遅延せられるトランザクション制御において、キャッシュデータの更新時に更新されたキャッシュ位置を記録する過程と、トランザクションの完了時に、前記記録から必要最小限の変更データのみをログファイルに書き出すことによりログデータ量を削減する過程とを含むように構成される。ここで、更新されたキャッシュ位置を記録する際に、その記録と先行する記録とをマージすることにより、ログファイルに書き出すログデータ量を最小化する過程を更に含むように構成することができる。

【0035】上述した本発明の第10の構成によれば、ログファイルに書き出されるログデータ量の削減が実現される。本発明の第11の構成は、本発明の第10の構成において、キャッシュが2次キャッシュを含むように構成される。

【0036】上述した本発明の第11の構成によれば、ログファイルを実ディスク上に書き出すログフラッシュ処理を、実行中のトランザクションと独立して行うことが可能となり、システム性能の向上が実現される。

【0037】

【発明の実施の形態】以下、本発明の実施の形態について詳細に説明する。図1は、本発明の実施の形態の構成を示すブロック構成図である。

【0038】#1～#3の各ノード101は、ファイル105が格納されているディスク装置と直結され、またローカルエリアネットワーク（LAN）106によって相互に接続される。

【0039】ファイル105を共用する複数のノード101（図中では、#1～#3）の全てにクライアント部102、そのうちの2つのノード101（図中では、#1と#2）にサーバ部103が存在する。

【0040】一方のノード101（#1）内のサーバ部103（#1）は主サーバ、他方のノード101（#2）のサーバ部103（#2）は従サーバと呼ばれる。それぞれのノード101内のクライアント部102は、主サーバであるノード101（#1）内のサーバ部103（#1）とのみ通信することにより、ファイル操作処理を実行する。

【0041】主サーバであるサーバ部103（#1）は、任意のクライアント部102からの要求（依頼）を処理して、その処理結果を、自身が保持するメタデータ104（#1）に反映させる。従サーバであるノード101（#2）内のサーバ部103（#2）が存在するときには、主サーバであるサーバ部103（#1）は、メタデータ104（#1）の更新内容（差分）をサーバ部103（#2）にも送る。従サーバであるサーバ部103（#2）は、送られてきたデータをノード101（#2）内のメタデータ104（#2）に反映させる。

【0042】任意のノード101内のクライアント部102は、図2に示されるように、そのノード101内のオペレーティングシステム（OS）201内に存在し、そのノード101内のユーザプログラム202からのファイル操作要求を、主サーバであるノード101（#1）内のサーバ部103（#1）の助けを借りて処理する。#1又は#2のノード101内のサーバ部103は、そのノード101内のオペレーティングシステム201に組み込んでもよいし、ユーザデーモンプログラムとしてオペレーティングシステム201の外に実装してもよい。このサーバ部103は、複数のノード101上のクライアント部102からのファイル操作要求を、LAN106（図1参照）を介して受け付ける。

【0043】上述の構成のもとでクライアント部102とサーバ部103がファイル操作制御を実行する場合、本実施の形態では、下記のトークンが用いられる。

1）ファイル105ごとに複数種類（例えば4種類）のトークンが用意され、その中に、ファイルサイズの拡張を制御しmultiple-read/single-write特性を有するサイズトークンが含ませられる。

2）ファイル105ごとに複数種類（例えば4種類）のトークンが用意され、その中に、ファイル時刻を制御しmultiple-write/multiple-read特性を有する時刻トークンが含ませられる。1つのノード101は、1つのファイル105について、read権の時刻トークンとwrite権の時刻トークンを同時に取得できる。ただし、或るノード101内のクライアント部102がサーバ部103に或るファイル105についてのread権の時刻トークンを要求したときに、他のノード101内のクライアント部102がそのファイル105についてのwrite権の時刻トークンを持っていた場合には、サーバ部103は、その、他のノード101内の時刻トークンを回収する。また逆に、或るノード101内のクライアント部102がサーバ部103に或るファイル105についてのwrite権の時刻トークンを要求したときに、他のノード101内のクライアント部102がそのファイル105についてのread権の時刻トークンを持っていた場合も、サーバ部103は、その、他のノード101内の時刻トークンを取り上げる。すなわち、1つのファイル105については、複数のノードがそれぞれ、そのファイル105についてのread権の時刻トークンとwrite権の時刻トークンを同時に保有するという事はない。

3）ファイル105ごとに複数種類（例えば4種類）のトークンが用意され、その中に、ファイルサイズの縮小を制御しmultiple-read/single-write特性を有する属性トークンが含ませられる。

4）ファイル105ごとに複数種類（例えば4種類）のトークンが用意され、その中に、ファイル内データのアクセス権を制御しファイル105を構成するブロックごとに存在するmultiple-read/single-write特性を有する

データトークンが含ませられる。また、本実施の形態は、下記の基本的動作を実行する。

5) 各トークンは、サーバ部103によって管理され、トークンが必要なノード101内のクライアント部102は、サーバ部103に、必要なトークンの獲得を要求(依頼)する。

6) サーバ部103は、ファイル105を格納するディスク上のどこが空いているかを示す空きブロック情報(空きエクステント情報)及び個々のファイル105のディスク上での存在場所(ファイル105のエクステント情報)を、メタデータ104として管理している。

7) クライアント部102は、サーバ部103に、ディスク上の空きブロック群(空きエクステント群)を事前要求(リザーブ要求)し、ユーザプログラム202からのwrite要求時には、事前要求で確保しておいた空きエクステント群の中から最適なものを割り当て、そこにユーザデータを書き込む。

続いて、本実施の形態の具体的な動作について、以下に順次説明する。

【0044】図3は、任意のノード101内のクライアント部102が実行するファイル操作要求制御のメイン動作フローチャートであり、図5及び図6は、主サーバであるノード101(#1)内のサーバ部103(#1)が実行するファイル操作要求制御のメイン動作フローチャートである。なお、以下の説明において、特に言及しない場合には、「サーバ部103」と記述した場合には、主サーバであるノード101(#1)内のサーバ部103(#1)を指すものとする。

1) クライアント部102及びサーバ部103でのopen操作処理

任意のノード101において、ユーザプログラム202(図2)がファイル105のopen要求を実行すると、同一のノード101内のクライアント部102がそのopen要求を受け取る(図3のステップ301の判定がYES)。この結果、クライアント部102は、open操作処理を実行する(図3のステップ302)。図4は、クライアント部102が実行する図3のステップ302のopen操作処理の動作フローチャートである。

【0045】まず、クライアント部102は、LAN106(図1)を介して、サーバ部103に、open要求を送信する。このopen要求には、アクセスの種別を示すオープンモード(read又はwrite)が付加される。

【0046】その後、クライアント部102は、サーバ部103からの応答を待つ(図4のステップ402→403→402の処理ループ)。なお、タイムアウト時には、クライアント部102は、エラー処理を実行し(図4のステップ403→404)、その後、図3のメイン動作フローチャートの処理ループに戻る。

【0047】サーバ部103は、クライアント部102からopen要求を受信すると(図5のステップ500の判

定がYES)、open操作処理を実行する(図5のステップ501)。図7は、サーバ部103が実行する図5のステップ501のopen操作処理の動作フローチャートである。

【0048】まず、サーバ部103は、受信されたopen要求によって指定されているファイル105(図1)について、そのopen要求によって指定されているオープンモードと矛盾するデータトークンを他のノード101に渡しているかどうかを調べる(図7のステップ701)。

【0049】サーバ部103は、上記オープンモードと矛盾するデータトークンを他のノード101に渡していない場合に、ファイル全体のデータトークン及びエクステント情報と、属性トークンと、サイズトークンと、時刻トークンと、属性データを、それぞれ応答データとして設定し(図7のステップ702~706)、応答処理を実行する(図7のステップ707)。ファイル全体のデータトークンとサイズトークンは、それぞれ、前記open要求によって指定されているオープンモードが、readならread権のトークン、writeならwrite権のトークンである。また、時刻トークンは、write権のトークンである。さらに、属性データには、例えばファイルサイズ、アクセス権、ファイル作成日付、ファイル更新日付等のデータが含まれる。

【0050】一方、サーバ部103は、上記オープンモードと矛盾するデータトークンを他のノード101に渡している場合には、ファイル全体のデータトークンは設定せずに、エクステント情報と、属性トークンと、サイズトークンと、時刻トークンと、属性データのみを、それぞれ応答データとして設定し(図7のステップ703~706)、応答処理を実行する(図7のステップ707)。

【0051】クライアント部102は、サーバ部103から応答を受信すると、その応答に含まれているファイル全体のデータトークン及びエクステント情報と、属性トークンと、サイズトークンと、時刻トークンと、属性データを、それぞれメモリ内のキャッシュ領域に保持する(図4のステップ402→405~409)。その後、クライアント部102は、ユーザプログラム202へのファイルディスクリプタの応答等の、その他のopen操作処理を実行し、その後、図3のメイン動作フローチャートの処理ループに戻る。

【0052】以上のようにして、本実施の形態では、ファイル105のopen時に、競合が発生していなければ、以降のファイルアクセス(readアクセス又はwriteアクセス)に必要なトークンが全て渡されるため、クライアント部102は、サーバ部103との間で、トークン獲得のための通信を行う必要が全くなくなるという効果を有する。

【0053】また、open要求時にファイル全体のトーク

ンが引き渡されることにより、可能な限り新たなトークン要求を行わずにファイルへの連続アクセスが可能となる。データベースアクセス等を除く一般的なファイルアクセスでは、1つのノード101からのwrite 要求の発行時に他のノード101からread命令が発行される確率は小さい。従って、1つのノード101に引き渡されたファイル全体のトークンが回収される確率も低く、ファイル105への連続アクセス時にアクセス単位ごとにトークン要求が不要になることによる性能向上が期待できる。

## 2) クライアント部102でのread操作処理

任意のノード101で、ユーザプログラム202がファイル105のread要求を発行すると、同一のノード101内のクライアント部102がそのread要求を受け取る(図3のステップ303の判定がYES)。この結果、クライアント部102は、read操作処理を実行する(図3のステップ304)。図8は、クライアント部102が実行する図3のステップ304のread操作処理の動作フローチャートである。

【0054】まず、クライアント部102は、必要な以下のトークンを保持しているかどうかを調べる(図8のステップ801)。

- ・read要求された範囲のread権のデータトークン
  - ・属性トークン
  - ・write 権の時刻トークン
  - ・read要求が最終ブロックのread要求である場合のみ、その最終ブロックについてのread権のサイズトークン
- ここで、属性トークンが存在すれば、ファイル105の最終ブロックの1つ前のブロックまではファイル内容が変更されていないことが保証されるため、かかるブロックのread操作処理時にはサイズトークンは獲得する必要はない。一方、read要求が最終ブロックのread要求である場合において、上記サイズトークンが存在しない場合には、他のノード101内のクライアント部102がその最終ブロックからのファイルサイズの拡張処理(write 操作処理)を実行している可能性があり、最終ブロックのread可能範囲が保証されない。上記サイズトークンが獲得された場合には、最終ブロックのread可能範囲が保証されるため、ユーザプログラム202は、その最終ブロックについてのread操作処理が可能となる。

【0055】このように本実施の形態では、ファイル105の最終ブロックにアクセスするのでなければ、サイズトークンを獲得することなくファイル105にアクセスすることが可能となり、これと並行して、他のノード101は、サイズトークンを獲得してファイル105の最終ブロックにアクセスし、ファイル105のサイズを拡張するwrite 操作処理を実行することができる。このため、例えばファイルを拡張するプログラムとファイルをその先頭から順に読むプログラムをそれぞれ異なるノード101で同時に実行させることが可能となり、シス

テム全体の性能を向上させることができる。

【0056】クライアント部102は、もし上記トークンを全て保持しているなら、サーバ部103にトークンを要求することなく、クライアント部102が保持する(キャッシュしている)データを使って、ユーザプログラム202の要求を処理する(図8のステップ801→802)。その後、クライアント部102は、図3のメイン動作フローチャートの処理ループに戻る。

【0057】一方、クライアント部102は、もし不足するトークンが存在するなら、そのトークンをLAN106(図1)を介してサーバ部103に要求し、サーバ部103からの応答を待つ(図8のステップ801→803、ステップ804→805→804の処理ループ)。なお、タイムアウト時には、クライアント部102は、エラー処理を実行し(図4のステップ403→404)、その後、図3のメイン動作フローチャートの処理ループに戻る。

【0058】クライアント部102は、サーバ部103から応答を受信すると、その応答に基づいてユーザプログラム202の要求を処理する(図8のステップ804→807)。その後、クライアント部102は、図3のメイン動作フローチャートの処理ループに戻る。

## 3) クライアント部102でのwrite 操作処理

任意のノード101で、ユーザプログラム202がファイル105のwrite 要求を発行すると、同一のノード101内のクライアント部102がそのwrite 要求を受け取る(図3のステップ305の判定がYES)。この結果、クライアント部102は、write 操作処理を実行する(図3のステップ306)。この処理は、read操作処理と同様の図8の動作フローチャートによって示される。

【0059】まず、クライアント部102は、必要な以下のトークンを保持しているかどうかを調べる(図8のステップ801)。

- ・write 要求された範囲のwrite 権のデータトークン
- ・属性トークン
- ・write 権の時刻トークン
- ・write 要求が最終ブロックのwrite 要求である場合のみ、その最終ブロックについてのwrite 権のサイズトークン

ここで、サイズトークンを用いることにより得られる効果は、read操作処理時の場合と同様である。

【0060】クライアント部102は、もし上記トークンを全て保持しているなら、サーバ部103にトークンを要求することなく、クライアント部102が保持する(キャッシュしている)データを使って、ユーザプログラム202の要求を処理する(図8のステップ801→802)。その後、クライアント部102は、図3のメイン動作フローチャートの処理ループに戻る。

【0061】一方、クライアント部102は、もし不足



するトークンが存在するなら、そのトークンをLAN106（図1）を介してサーバ部103に要求し、サーバ部103からの応答を待つ（図8のステップ801→803、ステップ804→805→804の処理ループ）。なお、タイムアウト時には、クライアント部102は、エラー処理を実行し（図4のステップ403→404）、その後、図3のメイン動作フローチャートの処理ループに戻る。

【0062】クライアント部102は、サーバ部103から応答を受信すると、その応答に基づいてユーザプログラム202の要求を処理する（図8のステップ804→807）。その後、クライアント部102は、図3のメイン動作フローチャートの処理ループに戻る。

4）クライアント部102でのファイル時刻操作処理  
任意のノード101において、ユーザプログラム202（図2）がファイル105に関するファイル時刻を要求すると、同一のノード101内のクライアント部102がその要求を受け取る（図3のステップ307の判定がYES）。この結果、クライアント部102は、ファイル時刻操作処理を実行する（図3のステップ308）。図9は、クライアント部102が実行する図3のステップ308のファイル時刻操作処理の動作フローチャートである。

【0063】まず、クライアント部102は、ユーザプログラム202から指定されたファイル105について、read権の時刻トークンのみを保持しているかどうかを調べる（図9のステップ901）。この判定がYESならば、クライアント部102は、自身が保持するファイル時刻をユーザプログラム202に回答する（図9のステップ903）。その後、クライアント部102は、図3のメイン動作フローチャートの処理ループに戻る。

【0064】上記判定がNOならば、クライアント部102は次に、ユーザプログラム202から指定されたファイル105について、read権とwrite権の各時刻トークンを保持しており、かつ前回サーバ部103から上記ファイル105に関するファイル時刻を取得してからそのファイル105に未アクセスであるかどうかを調べる（図9のステップ902）。この判定がYESの場合にも、クライアント部102は、自身が保持するファイル時刻をユーザプログラム202に回答する（図9のステップ903）。その後、クライアント部102は、図3のメイン動作フローチャートの処理ループに戻る。

【0065】上記ステップ903の判定もNOならば、クライアント部102は、LAN106を介してサーバ部103に、自クライアント部102でのそのファイル105に関するファイルアクセスの有無を付加した要求であって、read権の時刻トークンの獲得要求を送信する（図9のステップ904）。

【0066】その後、クライアント部102は、サーバ部103からの応答を待つ（図9のステップ905→

906→905の処理ループ）。なお、タイムアウト時には、クライアント部102は、エラー処理を実行し（図9のステップ906→907）、その後、図3のメイン動作フローチャートの処理ループに戻る。

【0067】クライアント部102は、サーバ部103からファイル時刻を受信すると、そのファイル時刻をユーザプログラム202に回答する（図9のステップ905→908）。また、クライアント部102は、そのファイル時刻を、クライアント部102内の上記ファイル105に対応するキャッシュ領域に保持する（図9のステップ909）。さらにクライアント部102は、上記キャッシュ領域において、上記ファイル105に対してファイルアクセスなしの状態を設定する（図9のステップ910）。

5）サーバ部103でのread権の時刻トークンの応答処理

任意のノード101において、クライアント部102が、前述した図3のステップ308及び図9のファイル時刻操作処理を実行することによって、サーバ部103にread権の時刻トークンを要求すると（図9のステップ904）、サーバ部103が、それを受け取ることにより（図5のステップ502の判定がYES）、read権の時刻トークンの応答処理を実行する（図5のステップ503）。図10は、サーバ部103が実行する図5のステップ503の応答処理の動作フローチャートである。

【0068】サーバ部103は、クライアント部102からread権の時刻トークンの獲得要求を受信すると、まずその時刻トークンに対応するwrite権の時刻トークンを保持するクライアント部102が存在するかどうかを調べる（図10のステップ1001）。

【0069】この判定がYESの場合は、クライアント部102は、上記write権の時刻トークンを保持する全てのクライアント部102に、そのwrite権の時刻トークンの回収要求を発行し、全てのクライアント部102からの応答を待つ（図10のステップ1001→1002、ステップ1003→1004→1003の処理ループ）。なお、タイムアウト時には、サーバ部103は、エラー処理を実行し（図10のステップ1004→1005）、その後、図5及び図6のメイン動作フローチャートの処理ループに戻る。

【0070】これに対して、各クライアント部102では、要求されたwrite権の時刻トークンの回収処理を実行する（図3のステップ309→310）。具体的には、各クライアント部102は、要求されたwrite権の時刻トークンを無効化すると共に、その時刻トークンに対応するファイル105に対するファイルアクセスの有無を、サーバ部103への応答に付加する。

【0071】サーバ部103は、ステップ1001の判定がNOであった場合、又は上記write権の時刻トークンを保持する全てのクライアント部102からの応答を



受信した場合に、read権の時刻トークンを要求しているクライアント部102に回答するファイル時刻を決定する(図10のステップ1006)。具体的には、要求元を含めて(図9のステップ904参照)、いずれかのノード101のクライアント部102がファイルアクセス有りを応答した場合は、サーバ部103は、自身がメタデータ104として保持する該当ファイル時刻を、現時刻により更新する。なお、各クライアント部102からファイルアクセス相対時刻間隔(何秒前にアクセスしたかを示すデータ)を応答させるようにし、応答された各クライアント部102からのファイルアクセス相対時刻間隔のうち最も小さい値によって、メタデータ104内の時刻を更新する(すなわち、[“現時刻” - “最も小さいファイルアクセス相対時刻間隔”]にする)ように構成されてもよい。一方、いずれのノード101もファイルアクセス無しを応答した場合は、サーバ部103は、自身が保持するメタデータ104中の該当ファイル時刻を、そのまま使用する。

【0072】続いて、サーバ部103は、決定したメタデータ104中のファイル時刻を、read権の時刻トークンを要求したクライアント部102に回答する(図10のステップ1007)。

【0073】最後に、サーバ部103は、要求元のクライアント部102にread権の時刻トークンを渡したことをサーバ部103の主記憶中に記憶する(図10のステップ1008)。

【0074】その後、サーバ部103は、図5及び図6のメイン動作フローチャートの処理ループに戻る。

6) サーバ部103でのwrite権の時刻トークンの応答処理

任意のノード101において、クライアント部102が、前述した図3のステップ304及び図8のread操作処理又は図3のステップ306及び図8のwrite操作処理を実行することにより、サーバ部103にwrite権の時刻トークンを要求すると、サーバ部103が、それを受け取ることにより(図5のステップ504の判定がYES)、write権の時刻トークンの応答処理を実行する(図5のステップ505)。図11は、サーバ部103が実行する図5のステップ505の応答処理の動作フローチャートである。

【0075】サーバ部103は、クライアント部102からwrite権の時刻トークンの獲得要求を受信すると、まずその時刻トークンに対応するread権の時刻トークンを保持するクライアント部102が存在するかどうかを調べる(図11のステップ1101)。

【0076】この判定がYESの場合は、クライアント部102は、上記read権の時刻トークンを保持する要求クライアント部102を除く全てのクライアント部102に、そのread権の時刻トークンの回収要求を発行し、全てのクライアント部102からの応答を待つ(図11

のステップ1101→1102, ステップ1103→1104→1103の処理ループ)。なお、タイムアウト時には、サーバ部103は、エラー処理を実行し(図11のステップ1104→1105)、その後、図5及び図6のメイン動作フローチャートの処理ループに戻る。

【0077】これに対して、各クライアント部102では、要求されたread権の時刻トークンの回収処理を実行する(図3のステップ309→310)。具体的には、各クライアント部102は、要求されたread権の時刻トークンを無効化し、サーバ部103に回答を返す。

【0078】サーバ部103は、ステップ1101の判定がNOであった場合、又は上記read権の時刻トークンを保持する全てのクライアント部102からの応答を受信した場合に、write権の時刻トークンを、要求クライアント部102に回答する(図11のステップ1106)。

【0079】最後に、サーバ部103は、要求元のクライアント部102にwrite権の時刻トークンを渡したことをメタデータ104中に記憶する(図11のステップ1107)。

【0080】その後、サーバ部103は、図5及び図6のメイン動作フローチャートの処理ループに戻る。上述の2)~6)で示したように、本実施の形態では、ユーザプログラム202がファイル105のread操作処理又はwrite操作処理を実行するときには、該当クライアント部102はそのファイル105についてのwrite権の時刻トークンを使用する。この際、クライアント部102はそのファイル105についてのwrite権の時刻トークンを保持していなければサーバ部103にそれを要求する。これに回答してサーバ部103は、他のノード101からそのファイル105に対応するread権の時刻トークンは回収するが、write権の時刻トークンは回収しない。従って、クライアント部102は、ユーザプログラム202が1つのファイル105に連続アクセスするような場合において、そのファイル105への最終的なアクセスが終了するまでwrite権の時刻トークンを返却する必要も、またアクセスの有無をサーバ部103に通知する必要もなく、他のノード101との間でそのファイル105のファイル時刻の同期をとる必要がなくなる。このため、システム全体の性能を向上させることが可能となる。

【0081】なお、上述の制御によると、write権の時刻トークンは、ユーザプログラム202がファイル105のファイル時刻を明示的に要求し、該当クライアント部102からサーバ部103にそのファイル105についてのread権の時刻トークンが要求された場合に回収されることになるが、これだけだと、ファイル時刻の要求が発生しない限り、ファイル105のファイル時刻がいつまでたってもサーバ部103側で確定しないことにな

る。これを防ぐために、例えば、クライアント部102は、ユーザプログラム202がファイル105をクローズしたタイミングで、サーバ部103にファイルアクセスの有無を通知し、サーバ部103はそれを受けてメタデータ104中の該当ファイル時刻を更新するように構成することができる。

7) サーバ部103でのデータトークンの応答処理  
任意のノード101において、クライアント部102が、前述した図3のステップ304及び図8のread操作処理又は図3のステップ306及び図8のwrite操作処理を実行することにより、サーバ部103にデータトークンを要求すると(図8のステップ803)、サーバ部103が、それを受け取ることにより(図5のステップ506の判定がYES)、データトークンの応答処理を実行する(図5のステップ507)。図12は、サーバ部103が実行する図5のステップ507の応答処理の動作フローチャートである。

【0082】サーバ部103は、クライアント部102からデータトークンの獲得要求を受信すると、まずその要求に矛盾するデータトークンを保持するクライアント部102が存在するかどうかを調べる(図12のステップ1201)。

【0083】この判定がYESの場合は、クライアント部102は、上記データトークンを保持する全てのクライアント部102に、そのデータトークンの回収要求を発行し、全てのクライアント部102からの応答を待つ(図12のステップ1201→1202、ステップ1203→1204→1203の処理ループ)。なお、タイムアウト時には、サーバ部103は、エラー処理を実行し(図12のステップ1204→1205)、その後、図5及び図6のメイン動作フローチャートの処理ループに戻る。

【0084】これに対して、各クライアント部102では、要求されたデータトークンの回収処理を実行する(図3のステップ309→310)。具体的には、各クライアント部102は、要求されたデータトークンを無効化し、サーバ部103に応答を返す。また、回収を要求されたデータトークンがwrite権のデータトークンである場合には、各クライアント部102は、そのwrite権のデータトークンで示されるファイル105の範囲で自身が更新したデータをキャッシュからディスク上に書き戻し、新たにそのファイル105に割り当てたエクステント情報を、上記応答に付加する。

【0085】サーバ部103は、上述のデータトークンを保持する全てのクライアント部102からの応答を受信した場合に、上記応答がwrite権のデータトークンに関するものであるならば、応答されたファイル105のエクステント情報を、自身が保持するメタデータ104に反映させる(図12のステップ1203→1206)。

【0086】その後、サーバ部103は、要求元のクライアント部102から指定された範囲のエクステント情報が付加されたデータトークンを、上記クライアント部102に応答する(図12のステップ1207)。

【0087】一方、クライアント部102からのデータトークンの獲得要求に矛盾するデータトークンを保持するクライアント部102が存在せずステップ1201の判定がNOで、かつファイル全体のデータトークンを応答しても競合が発生せずステップ1208の判定もNOである場合には、サーバ部103は、ファイル全体のエクステント情報とファイル全体のデータトークンを、要求元のクライアント部102に応答する(図12のステップ1201→1208→1209)。

【0088】上記競合が発生する場合には、サーバ部103は、要求元のクライアント部102から指定された範囲のエクステント情報が付加されたデータトークンを、上記クライアント部102に応答する(図12のステップ1207)。

【0089】ステップ1207又は1209の処理の後、サーバ部103は、図5及び図6のメイン動作フローチャートの処理ループに戻る。サーバ部103からデータトークンを取得したクライアント部102は、前述した図4のステップ405又は図8のステップ807の処理において、自身が該当ファイル105に対応するデータトークンを保持していること、及び応答されたエクステント情報を、メモリ内のキャッシュ領域に記憶する。そして、クライアント部102は、それ以降のユーザプログラム202からの要求に基づくファイルアクセス処理(図8のステップ802)は、上記エクステント情報で示される、ディスク上のブロックに対して実行する。

【0090】上述したように、データトークンの応答時に、ファイル105のエクステント情報も同時に応答される。このため、複数のノード101は、ディスク装置内のファイル105に、LAN106経由ではなく直結された制御・データ線を介してアクセスすることが可能となる。

8) サーバ部103におけるサイズトークンの応答処理  
サーバ部103は、クライアント部102からサイズトークンを要求された場合には、その要求と矛盾するサイズトークンを他のクライアント部102から回収した上で、要求されたサイズトークンにファイルサイズを付加して要求元のクライアント部102に応答する(図5のステップ506→507)。その後、サーバ部103は、図5及び図6のメイン動作フローチャートの処理ループに戻る。

9) サーバ部103における属性トークンの応答処理  
サーバ部103は、クライアント部102から属性トークンを要求された場合には、その要求と矛盾する属性トークンを他のクライアント部102から回収した上で、

要求された属性トークンにファイル属性を付加して要求元のクライアント部102に应答する(図5のステップ508→509)。その後、サーバ部103は、図5及び図6のメイン動作フローチャートの処理ループに戻る。

#### 10) エクステント管理の詳細

次に、サーバ部103及びクライアント部102におけるエクステント(ディスク領域)の管理の詳細について説明する。

【0091】まず、サーバ部103は、複数のディスクボリュームを管理することができ、メタデータ104として、ファイル105の属性データ、各ディスクボリューム毎の空きエクステントに関する情報(空きスペース情報)、及びクライアント部102に貸し出したエクステントに関する情報(リザーブスペース情報)を保持している。

【0092】空きスペース情報とリザーブスペース情報は、図13に示されるように、空きスペースBツリー1301として管理され、そのうち空きスペース情報は空きスペースキュー1302からアクセスでき、リザーブスペース情報はリザーブスペースキュー1303からアクセスできる。

【0093】空きスペースキュー1302は、ディスクボリューム毎に、空きスペースBツリー1301に接続されている使用可能エクステント(使用中でもリザーブ中でもないエクステント)を管理する。

【0094】リザーブスペースキュー1303は、クライアント部102毎に、そのクライアント部102にリザーブされ空きスペースBツリー1301に接続されているエクステントを管理する。

【0095】また、サーバ部103は、使用中のエクステントは、iノードBツリー1304によって管理する。一方、クライアント部102は、サーバ部103に要求することによりリザーブしたエクステントを、リザーブキュー1305によって管理する。

【0096】クライアント部102は、主記憶上にキャッシュを持ち、ユーザプログラムが要求したディスク上のデータをキャッシュする。サーバ部103内の空きスペースキュー1302とクライアント部102内のリザーブキュー1305は、ディスクボリューム毎に予め決められた個数分のヘッダを有しており、各ヘッダがエクステントのサイズに対応している。例えば、ヘッダの個数を4個とすると、各ヘッダが、1~4KB(キロバイト)、4~16KB、16~64KB、64~256KBの各サイズ範囲のエクステント群(空きスペースBツリー1301)を管理する。ヘッダの個数と各ヘッダが表すサイズは、各ディスクボリュームのファイルシステムを作成したときに決定される。

【0097】図14は、1つのノード101(図1参照)内において、ユーザプログラム202(図2参照)

が、ファイル105へのデータ書き込み(write 要求)を依頼したときのエクステント管理のシーケンスを示す図である。このシーケンスにおいて、クライアント部102が実行する処理は、図3のステップ306のwrite 操作処理における図8のステップ807の処理の一部である。また、サーバ部103が実行する処理は、図5のサーバ部103のメイン動作フローチャート内の特には図示しない一部の処理である。

【0098】図14において、ユーザプログラム202がファイル105に対するwrite 要求を発行すると、クライアント部102は、キャッシュにデータを保持する。ユーザプログラム202がファイル105をクローズし、又はキャッシュが一杯になり、或いはサーバ部103からデータトークンの回収を要求される(図12のステップ1202参照)ことにより、キャッシュされているデータをディスクに書き出す必要が発生した場合に、クライアント部102は、サーバ部103から受け取っていたファイル105のエクステント情報(図4のステップ405参照)を調べ、その要求が既にディスク領域が割り当てられているファイル領域に対するものであるか否かを認識し、ファイル105毎にキャッシュ内でエクステントが割り当てられていない領域で隣接するものをまとめる(このまとめられたファイル領域を書出し対象領域と呼ぶ)。次に、クライアント部102は、書出し対象領域のサイズを調べると共に、その領域の性質に従って、以下の何れかの処理を実行する。

①書出し対象領域に隣接する(直前の)領域に、同じファイル105に関するエクステントが既にサーバ部103から割り当てられている場合：クライアント部102は、割り当てられているエクステントのブロックアドレスと書出し対象領域のサイズを指定して、それに続くエクステントのリザーブ(貸し出し)をサーバ部103に依頼し、応答されたエクステントにデータを書き込む。なお、サーバ部103は、依頼されたエクステントが既に割当て済みの場合には、他のエクステントを返す。

②書出し対象領域に隣接する(直前の)領域に、同じファイル105に関するエクステントがいまだサーバ部103から割り当てられていない場合：クライアント部102は、書出し対象領域のサイズに対応するリザーブキュー1305の先頭に接続されているエクステントにデータを書き出す。クライアント部102は、リザーブキュー1305から、そのエクステントを取り除く。

以上の動作の後、クライアント部102は、サーバ部103に書出し完了を通知する。この際、クライアント部102は、使用したエクステント(リザーブスペース)のアドレスと、書出し対象領域のサイズを通知する。

【0099】サーバ部103は、通知されたエクステント(リザーブスペース)のアドレスと、書出し対象領域のサイズとから、メタデータ104内の対象ファイル105に関する属性データを更新し、リザーブスペースキ

ユー1303及び空きスペースBツリー1301上から、クライアント部102から通知されたエクステントを取り除き、そのエクステントを1ノードBツリー1304に接続する。書き出されたエクステントのサイズが使用されたリザーブスペースよりも小さい場合には、サーバ部103は、残りのエクステントを、空きスペースとして空きスペースキュー1302の当該エクステントのサイズに対応するヘッダに接続する。

#### 11) エクステント群のリザーブ制御処理

クライアント部102は、一定時間が経過するごとに、エクステント群リザーブ要求処理を実行する(図3のステップ311→312)。この処理では、クライアント部102は、自身がリザーブキュー1305にリザーブしているエクステント群を調べ、リザーブ量が一定値以下になった場合に、サーバ部103に一定個数のエクステント群のリザーブを要求する。この処理は、各サイズのヘッダ毎に行われ、不足が発生したヘッダ以外についても、各リザーブ量が所定値以上となるように、各ヘッダに対して上記リザーブ処理が実行される。

【0100】サーバ部103は、エクステント群のリザーブ要求を受信すると、エクステント群のリザーブ処理を実行する(図6のステップ512→513)。この処理では、サーバ部103は、空きスペースキュー1302に接続されている空きスペースBツリー1301中から、使用可能なエクステント群を探し、それらを空きスペースキュー1302からリザーブスペースキュー1303に繋ぎ替えた後に、そのリザーブしたエクステント群をクライアント部102に応答する。その後、サーバ部103は、図5及び図6のメイン動作フローチャートの処理ループに戻る。

【0101】クライアント部102は、図3のステップ312において、サーバ部103から応答されたエクステント群をリザーブキュー1305に繋ぎ、ステップ312を終了して、図3のメイン動作フローチャートの処理ループに戻る。

【0102】サーバ部103は、自身に対してmountを行っているクライアント部102の障害を検出した場合、又はクライアント部102からunmount要求を受信した場合には、そのクライアント部102に対してリザーブしていたリザーブスペースキュー1303中のエクステント群の解放処理を実行して、それらを空きスペースキュー1302に繋ぎ替える(図5のステップ514→515)。その後、サーバ部103は、図5及び図6のメイン動作フローチャートの処理ループに戻る。

【0103】上述のように、本実施の形態では、空きエクステント群がリザーブされることにより、クライアント部102は、サーバ部103に問い合わせることなく、キャッシュを活用して新たなエクステントをファイル105に割り当てることが可能となる。このため、クライアント部102とサーバ部103との間の通信回数

を削減でき、システム全体の性能を向上させることが可能となる。

【0104】また、新たに割り当てられたエクステントは、データが書き込まれた後のクライアント部102からサーバ部103への応答によって初めて、そのファイル105のメタデータ104として記憶される。このため、悪意をもってデータを覗くことを防止することが可能となる。

#### 12) 主サーバと従サーバの同期処理

主サーバであるノード101(#1)内のサーバ部103(#1)は、例えば図7、図10、図11、図12などにおいて、メタデータ104(#1)を更新する場合は、従サーバであるノード101(#2)内のサーバ部103(#2)に対して、メタデータ変更分と時刻データを送信し、従サーバがそれらを受信したことを確認した後に、クライアント部102に応答を返す。

【0105】従サーバであるノード101(#2)内のサーバ部103(#2)は、上述のメタデータ変更分と時刻データを受信すると、メタデータ変更分を自身のメタデータ104(#2)に反映させると共に、送られてきた時刻データを記憶する(図6のステップ516→517)。その後、サーバ部103(#2)は、図5及び図6のメイン動作フローチャートの処理ループに戻る。

#### 13) 主サーバにおける障害発生時の、従サーバへの切替処理

従サーバであるノード101(#2)内のサーバ部103(#2)は、主サーバであるノード101(#1)内のサーバ部103(#1)の障害を監視しており、その障害を検出した場合には、サーバ切替処理を実行する(図6のステップ518→519)。このとき、サーバ部103(#2)は、最後に主サーバであるサーバ部103(#1)から送られてきた時刻を過ぎるまで、自身の時刻の待ち合せを実行する。その後、サーバ部103(#2)は、図5及び図6のメイン動作フローチャートの処理ループに戻る。

【0106】上述の制御により、サーバ切替時にも、矛盾のないファイル時刻の付与が可能となる。次に、上述したようなノード間ファイル共用管理システムにおいて、分散ファイルシステムの耐故障性を高めるためのログ制御機構を実現するための実施の形態について説明する。

【0107】図15は、ログ制御機構を実装したノード間ファイル共用管理システムの基本構成図である。共用ファイル管理装置1501(図1のサーバ部103を有するノード101に対応する)は、共用されるファイルの「属性」や「実ディスク上での格納位置」などの、ファイルごとに存在する制御情報(ファイル情報と呼ぶ)と、実ディスクの空き領域などを示す制御情報(ディスク情報と呼ぶ)を保持している。これら2つの管理情報

を総称してメタデータ1502（図1のメタデータ104に対応する）と呼び、障害に備えディスク上に格納されている。

【0108】共用ファイル管理装置1501は、データを共用する#1～#nの各ノード1503（クライアント部102を有するノード101に対応する）からの要求に従い、メタデータ1502をディスクから読み込み或いは更新し、ファイル情報を応答として返す。この際、異なる複数のメタデータブロックがアクセスされる可能性がある。

【0109】各ノード1503は、返されたファイル情報をメモリ上にキャッシュし、それ以降必要が生ずるまで、共用ファイル管理装置1501と通信することなく、キャッシュされたメモリ上のファイル情報のみを用いて処理を実行する。

【0110】各ノード1503がそれぞれのキャッシュ上に保持するファイル情報相互間の一貫性を保証するために、トークンが使用される。トークンは、ファイル情報がノード1503に返される際に共用ファイル管理装置1501によりそのノード1503に対して発行され、共用ファイル管理装置1501が或るノード1503から矛盾する要求を受け付けたときに共用ファイル管理装置1501によって必要なノード1503から回収される。

【0111】回収を指示されたノード1503は、トークンによって指示されるキャッシュデータを無効化し、他ノード1503に伝えられるべき自身が行なったファイル情報の変更を応答する。

【0112】応答を受けた共用ファイル管理装置1501は、通知された変更をメタデータ1502に反映した後に、要求に基づく処理を再開し、要求元に対して結果を応答すると共にトークンを発行する。

【0113】共用ファイル管理装置1501が各ノード1503からの要求を処理するためには、メタデータ1502へのアクセスが必要となる。この場合に、毎回ディスクをアクセスしていたのでは性能が悪くなる。このため、ディスク上のデータを保持するバッファキャッシュ1504が共用ファイル管理装置1501内に設けられ、ディスクアクセスが削減される。バッファキャッシュ1504は、ディスク上の各ブロックに対応したエントリを持ち、各エントリにそのエントリのロックの有無を表示するためのロックワードが用意されることにより、或るスレッドが更新中のデータを他の要求を処理している他のスレッドが参照することが抑止される。

【0114】メタデータ1502の実ディスクへの反映は、要求処理が全て正常に終了した時点、いわゆるトランザクション完了時まで遅らされる。トランザクションが正常に終了すると、バッファキャッシュ1504上に保持されている更新データが一括してログファイル1505に書き出され、その後、更新データのディスクへの

反映タイミングがスケジュールされる。

【0115】ログファイル1505はサイクリックに使用され、実ディスクへの書き込みが完了するたびに、書き出しが完了した変更を保持するログ領域は空き領域に戻される。従って、実ディスクへの書き出しがまだ完了していない、成功した要求に伴うメタデータの変更は必ずログファイル1505上に存在するので、共用ファイル管理装置1501で障害が発生しても、メタデータ1502の復旧は容易にかつ高速に行なえるという特徴を有する。

【0116】次に、本実施の形態に係る上記基本構成に基づくロック継承制御処理につき、図16の説明図に基づいて説明する。尚、複数のクライアントから発行れる同一ファイルに対する操作要求を逐次化するためのファイル管理装置1501はファイル毎に用意するファイルロックを使用する。

【0117】本実施の形態では、1つのノード1503からの要求を処理するために共用ファイル管理装置1501上で実行される第1の実行単位（スレッド）は、他のノード1503に発行しているトークンを回収する場合に、トークン処理の対象となっているファイルを示す情報を保持したトークン回収制御表1602をトークン回収待ちキュー1601につなぎ、該当するノード1503に対してトークン回収要求を送信した後、トークン回収完了メッセージの到着を待ち合わせる。

【0118】トークンを保持しているノード1503におけるキャッシュの無効化が完了しそこから共用ファイル管理装置1501（図15）にトークン回収完了メッセージが通知されると、トークン回収完了メッセージを処理するために共用ファイル管理装置1501上で実行される第2の実行単位（スレッド）が、トークン回収待ちキュー1601を調べ、そのメッセージに対応するトークン回収制御表1602がキュー上に存在するならば、その制御表に「ロック継承中」を表示した上で、メタデータ1502（図15）の更新処理及びトークンの解放処理を実行する。

【0119】トークン回収完了メッセージの到着を待ち合わせていた第1の実行単位の待ちは、第2の実行単位によるトークン解放処理の結果解かれる。各ノード1503は、共用ファイル管理装置1501からの要求に基づかずに自律的に、トークン回収完了メッセージを共用ファイル管理装置1501に通知することもできる。従って、トークン回収完了メッセージが共用ファイル管理装置1501に到着した際に、トークン回収待ちキュー1601に該当するトークン回収制御表1602がつかっていない場合が起こり得る。このようなときには、上記第2の実行単位は、通常のファイルロック獲得処理を実行し、この結果他の実行単位がファイルロックを保持していればファイルロックの解放を待ち合わせ、ファイルロックがはずれたらメタデータの更新処理及びトー

クン解放処理を実行する。

【0120】上記第1の実行単位は、複数のノード1503に対してトークン回収要求を送信する可能性がある。このような場合には、共用ファイル管理装置1501は、複数のノード1503からトークン回収完了メッセージを相次いで受信する可能性がある。上記第2の実行単位は、第1番目のトークン回収完了メッセージを受信した時点で該当するトークン回収制御表1602にロック継承中を表示する。そして、第2番目以降のトークン回収完了メッセージを受信した他の各実行単位は、対応するトークン回収制御表1602にロック継承中が表示されていた場合には、継承中表示がオフとなるのを待ち合わせ、待ちが解けた時点でメタデータの更新処理及びトークン解放処理を実行する。このように、ロックの継承を行うことのできる実行単位は1つに制限される。

【0121】以上のロック継承制御により、トークン制御において、デッドロックの発生を回避することのできる効率的なファイルロック制御が実現される。次に、本実施の形態に係る図15に示される基本構成に基づくデッドロック検出処理について、図17の説明図に基づき説明する。

【0122】共用ファイル管理装置1501（図15）は、各ファイルを管理するファイル制御表1701に、ファイルロックワード1701aに対応して、そのファイルロックを保持している実行単位（スレッド）を示すオーナ1701bを設定し、また、各バッファキャッシュ1504（図15）のエントリを管理するバッファキャッシュ制御表1702に、バッファキャッシュロックワード1702aに対応して、そのバッファキャッシュロックを保持している実行単位（スレッド）を示すオーナ1702bを設定する。

【0123】また、共用ファイル管理装置1501は、各実行単位（スレッド）を管理するスレッド制御表1703に、その実行単位が待ち合わせしている対象を特定する情報である待ちリソース1703aと、その待ち合わせの原因を示す情報であるタイプ1703bを設定する。待ちリソース1703aとタイプ1703bには下記の何れかの設定が行われる。

1. ファイルロックの解放を待ち合わせる場合：

・タイプ1703bには、ファイルロック待ちを設定。

【0124】・待ちリソース1703aには、該当するファイル制御表1701内のファイルロックワード1701aを指示する情報を設定。

2. バッファキャッシュロックの解放を待ち合わせる場合：

・タイプ1703bには、バッファキャッシュロック待ちを設定。

【0125】・待ちリソース1703aには、該当するバッファキャッシュ制御表1702内のバッファキャッシュロックワード1702aを指示する情報を設定。

3. トークン回収を待ち合わせる場合：

・タイプ1703bには、トークン回収待ちを設定。

【0126】・待ちリソース1703aには、該当するファイルを指示する情報を設定。

以上の情報を使い、各スレッド（実行単位）は、以下のようにデッドロックを検出する。

＜スレッド（以下、スレッドAという）がファイルロックを要求した場合＞

ステップ1：スレッドAは、ファイルロックの解放待ちに入る前に、そのファイルに対応するファイル制御表1701内のファイルロックワード1701aとオーナ1701aとから、そのファイルロックを保持しているスレッド（以下、スレッドBという）に対応するスレッド制御表1703を取得する。

ステップ2：スレッドAは、そのスレッド制御表1703内の待ちリソース1703aとタイプ1703bとから、スレッドBが待ち合わせている資源を求める。スレッドBが待ち合わせている資源がないかスレッドBがトークン回収を待ち合わせているならば、スレッドAは、デッドロックは発生していないと判定し、ファイルロックの解放待ちに入る。

ステップ3：スレッドBがトークン回収の待ち合わせ以外の待ち合わせをしている場合には、スレッドAは、スレッドBが待ち合わせている資源に対するロックを保持しているスレッドを求める。

ステップ4：スレッドAは、ステップ3で求めたスレッドがスレッドA自身ならば、デッドロックが発生したと判定し、スレッドA自身が実行しているトランザクションをキャンセルする。そうでなければ、スレッドAは、ステップ2の処理を繰り返す。

＜スレッドAがバッファキャッシュロックを要求した場合＞

ステップ1：スレッドAは、バッファキャッシュロックの解放待ちに入る前に、そのバッファキャッシュエントリに対応するバッファキャッシュ制御表1702内のバッファキャッシュロックワード1702aとオーナ1702bとから、そのバッファキャッシュロックを保持しているスレッドBに対応するスレッド制御表1703を取得する。

ステップ2：スレッドAは、そのスレッド制御表1703内の待ちリソース1703aとタイプ1703bとから、スレッドBが待ち合わせている資源を求める。スレッドBが待ち合わせている資源がないならば、スレッドAは、デッドロックは発生していないと判定し、バッファキャッシュロックの解放待ちに入る。

ステップ3：スレッドAは、スレッドBが待ち合わせている資源がトークン回収待ちという資源で且つトークン回収待対象ファイルのファイルロックをスレッドAが保持しているならば、デッドロックが発生したと判定する。



ステップ4：スレッドAは、スレッドBが待ち合わせている資源に対するロックを保持しているスレッドを求める。

ステップ5：スレッドAは、ステップ4で求めたスレッドがスレッドA自身ならば、デッドロックが発生したと判定し、スレッドA自身が実行しているトランザクションをキャンセルする。そうでなければ、スレッドAは、ステップ2の処理を繰り返す。

以上説明したデッドロックの検出処理により、トークンに基づいてトランザクション制御されているメタデータ1502等の更新処理におけるデッドロックの発生を適切に検出することができる。

【0127】次に、本実施の形態に係る図15に示される基本構成に基づくログファイルの2次キャッシュ制御処理につき、図18の説明図に基づいて説明する。2次キャッシュ1801は、ログファイル1505（図15）には書出しが完了しているが、ディスクへの反映は完了していないメタデータ1502を保持するキャッシュで、トランザクションキャンセル時の性能劣化の防止、通常処理での性能向上を図るために、共用ファイル管理装置1501上に設けられる。

【0128】トランザクションが正常終了した場合、バッファキャッシュ1504上で更新されたデータは2次キャッシュ1801に送られ、変更表示がオンされる。ログファイル1505の空き領域が不足してくると、2次キャッシュ1801上の変更表示がオンになっているデータが実ディスクに書き出され、変更表示がリセットされる。

【0129】バッファキャッシュ1504から2次キャッシュにデータが移動させられる際に、2次キャッシュ1801の空き領域がなければ、変更表示がオンされていない2次キャッシュ領域が再使用される。

【0130】もし、全てのページの変更表示がオンされているならば、一定の量の変更されたページが実ディスクに書き出され、変更表示がオフにさせられた後に再使用される。

【0131】必要なメタデータ1502がバッファキャッシュ1504上に存在しない場合には、2次キャッシュ1801にデータが存在するならばそのデータが2次キャッシュ1801からバッファキャッシュ1504にコピーされる。必要なデータが2次キャッシュ1801にも存在しない場合には、そのデータがディスクからバッファキャッシュ1504に読み込まれる。

【0132】以上説明した2次キャッシュ制御処理により、バッファキャッシュ1504の変更内容を実ディスク上に書き出すログフラッシュ処理を、実行中のトランザクションと独立して行うことが可能となり、システム性能の向上が実現される。

【0133】続いて、本実施の形態に係る図15に示される基本構成に基づく、ログデータ量を削減できるログ

制御処理につき、図19の説明図に基づいて説明する。メタデータ1502がバッファキャッシュ1504上で更新された場合に、スレッドごとに存在するログキュー1901に、更新されたメタデータ1502の範囲を示す情報を記憶したログ制御表1902が追加される。この情報は、図19に示されるように、バッファキャッシュ1504上のエントリを指示するエントリIDと、そのエントリに属する範囲の始点アドレスstartと終点アドレスendとからなる。

【0134】この際、ログキュー1901がサーチされ、ログキュー1901上に、更新されたメタデータ1502の範囲に対してオーバラップするか隣接する範囲を表すログ制御表1902が既に存在するならば、旧ログ制御表1902の範囲が変更させられるだけで、新しいログ制御表1902は作成されない。

【0135】トランザクションが正常に終了した場合、ログキュー1901上のログ制御表1902から、変更されたメタデータ1502が認識され、それがログファイル1505にログデータとして書き出される。書出しが完了したら、該当するバッファキャッシュ1504のエントリに対するロックが解放される。

【0136】トランザクションが失敗に終わった場合には、ログキュー1901から更新されたメタデータ1502が認識され、該当するバッファキャッシュ1504上のエントリが無効化される。

【0137】以上説明したログ制御処理により、ログファイル1505に書き出されるログデータ量の削減が実現される。最後に、本実施の形態に係る図15に示される基本構成に基づく、トランザクションキャンセル時におけるメモリ常駐制御表のリストア制御処理につき、図20の説明図に基づいて説明する。

【0138】トランザクション処理の途中でデッドロック条件が検出されたり要求元のエラーなどが検出されることによりトランザクションがキャンセルされる場合には、バッファキャッシュ1504（図15）の無効化が行なわれる。これと共に、スレッドごとに存在するファイルロックキュー2001に接続されている各ファイル制御表2002がサーチされることにより、トランザクションの過程で獲得され解放されていないファイルロックが、全て解放させられる。

【0139】ここで、ファイル制御表2002には、ファイルロックの獲得に伴って、共用ファイル管理装置1501（図15）内のメモリ上に存在する常駐制御表2003が書き換えられた場合に、その更新を示す制御表更新フラグが設定される。なお、1つのファイル制御表2002には、複数の常駐制御表2003に対応する複数の制御表更新フラグを、制御表更新マップとして設定することができる。

【0140】今、トランザクションのキャンセルに伴いファイルロックが解除される際に、それに対応するファ



イルロックワードが設定されていたファイル制御表2002において何れかの制御表更新フラグがオンになっている場合には、ファイルロックの再獲得時にその制御表更新フラグに対応する常駐制御表2003のリロードが必要なことを示すリロードインジケータ(複数可)が表示された上で、ファイルロックが解放させられる。

【0141】トランザクションがデッドロック検出等によりキャンセルされた場合には、その後、そのトランザクションに対応する要求が始めから再試行される。そして、ファイルロックの再獲得時に、それに対応するファイルロックワードが設定されていたファイル制御表2002に何れかのリロードインジケータが表示されているならば、ファイルロックの獲得後に上記リロードインジケータによって指示される常駐制御表2003が、メタデータ1502(図15)の情報を使ってメモリ上に再構築される。

【0142】以上説明したリストア制御処理により、トランザクションのキャンセルに伴う常駐制御表2003の高速なリストアが実現される。ここで、本発明は、コンピュータにより使用されたときに、上述の本発明の実施の形態によって実現されるクライアント部102の機能又はサーバ部103の機能と同様の機能をコンピュータに行わせるためのコンピュータ読出し可能記録媒体として構成することもできる。この場合に、例えばフロッピーディスク、CD-ROMディスク、光ディスク、リムーバブルハードディスク等の可搬型記録媒体や、ネットワーク回線経由で、本発明の実施の形態の各種機能を実現するプログラムが、ノードを構成するコンピュータの本体内のメモリ(RAM又はハードディスク等)にロードされて、実行される。

#### 【0143】

【発明の効果】本発明の第1の態様の構成によれば、例えばopen要求時等においてファイル全体のトークンが引き渡されることにより、可能な限り新たなトークン要求を行わずにファイルへの連続アクセスが可能となる。データベースアクセス等を除く一般的なファイルアクセスでは、1つのノードからのwrite要求の発行時に他のノードからread命令が発行される確率は小さい。従って、1つのノードに引き渡されたファイル全体のトークンが回収される確率も低く、ファイルへの連続アクセス時にアクセス単位ごとにトークン要求が不要になることによる性能向上が期待できる。

【0144】本発明の第2の態様の構成によれば、クライアント装置は、ユーザプログラムが1つのファイルに連続アクセスするような場合において、そのファイルへの最終的なアクセスが終了するまでwrite権の時刻トークンを返却する必要もまたアクセスの有無をサーバ部に通知する必要もなく、他のノードとの間でそのファイルのファイル時刻の同期をとる必要がなくなる。このため、システム全体の性能を向上させることが可能とな

る。

【0145】本発明の第3の態様の構成によれば、ファイルの最終ブロックにアクセスするのであれば、サイズトークンを獲得することなくファイルにアクセスすることが可能となり、これと並行して、他のノードは、サイズトークンを獲得してファイルの最終ブロックにアクセスし、ファイルのサイズを拡張するwrite操作処理を実行することができる。このため、例えばファイルを拡張するプログラムとファイルをその先頭から順に読むプログラムをそれぞれ異なるノードで同時に実行させることが可能となり、システム全体の性能を向上させることができる。

【0146】本発明の第4の態様の構成によれば、複数のノードは、ディスク装置内のファイルに、LAN経由ではなく直結された制御・データ線を介してアクセスすることが可能となる。

【0147】本発明の第5の態様の構成によれば、サーバ切替時にも、矛盾のないファイル時刻の付与が可能となる。本発明の第6の態様の構成によれば、クライアント装置は、サーバ装置に問い合わせることなく、新たなブロックをファイルに割り当てることが可能となる。このため、クライアント装置とサーバ装置との間の通信回数を削減でき、システム全体の性能を向上させることが可能となる。更に、サイズ毎にリザーブすることにより最適な連続ブロックを割り当てフラグメンテーションを防止すると共にファイルアクセス性能を向上させることができる。また、新たに割り当てられたエクステン트는、データが書き込まれた後のクライアント装置からサーバ装置への応答によって初めて、そのファイルのメタデータ等として記憶される。このため、悪意をもってデータを覗くことを防止することが可能となる。

【0148】本発明の第7の構成によれば、トークン制御において、デッドロックの発生を回避することのできる効率的なファイルロック制御が実現される。本発明の第8の構成によれば、トークンに基づいてトランザクション制御されているメタデータ等の更新処理におけるデッドロックの発生を適切に検出することができる。

【0149】本発明の第9の構成によれば、トランザクションのキャンセルに伴う常駐制御表の高速なリストアが実現される。本発明の第10の構成によれば、ログファイルに書き出されるログデータ量の削減が実現される。

【0150】本発明の第11の構成によれば、ログファイルを実ディスク上に書き出すログフラッシュ処理を、実行中のトランザクションと独立して行うことが可能となり、システム性能の向上が実現される。

#### 【図面の簡単な説明】

【図1】本発明の実施の形態のシステム構成図である。

【図2】ノード内のソフトウェア構成図である。

【図3】クライアント部のメイン動作フローチャートで

ある。

【図4】クライアント部のopen操作処理の動作フローチャートである。

【図5】サーバ部のメイン動作フローチャート（その1）である。

【図6】サーバ部のメイン動作フローチャート（その2）である。

【図7】サーバ部のopen操作処理の動作フローチャートである。

【図8】クライアント部のread/write操作処理の動作フローチャートである。

【図9】クライアント部のファイル時刻操作処理の動作フローチャートである。

【図10】サーバ部でのread権の時刻トークンの応答処理の動作フローチャートである。

【図11】サーバ部でのwrite権の時刻トークンの応答処理の動作フローチャートである。

【図12】サーバ部でのデータトークンの応答処理の動作フローチャートである。

【図13】エクステント管理の詳細を示す図である。

【図14】エクステント管理のシーケンス図である。

【図15】ログ制御機構を実装したノード間ファイル共有管理システムの基本構成図である。

【図16】ロック継承制御処理の説明図である。

【図17】デッドロック検出処理の説明図である。

【図18】ログファイルの2次キャッシュ制御の説明図である。

【図19】ログデータ量を削減できるログ制御処理の説明図である。

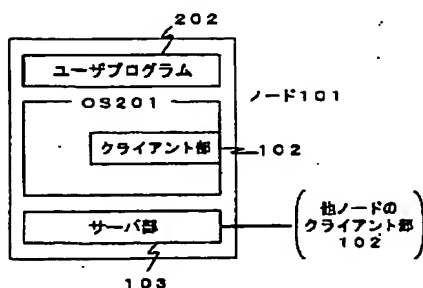
【図20】トランザクションキャンセル時におけるメモリ常駐制御表のリスト処理の説明図である。

【符号の説明】

101、1503	ノード
102	クライアント部
103	サーバ部
104、1502	メタデータ
105	ファイル
106	LAN
201	オペレーティングシステム（OS）
202	ユーザプログラム
1501	共用ファイル管理装置
1504	バッファキャッシュ
1505	ログファイル
1601	トークン回収待ちキュー
1602	トークン回収制御表
1701、2002	ファイル制御表
1701a、1702a	ファイルロック
1701b、1702b	オーナー
1702	バッファキャッシュ制御表
1703	スレッド制御表
1703a	待ちリソース
1703b	タイプ
1801	2次キャッシュ
1901	ログキュー
1902	ログ制御表
2001	ファイルロックキュー
2003	常駐制御表

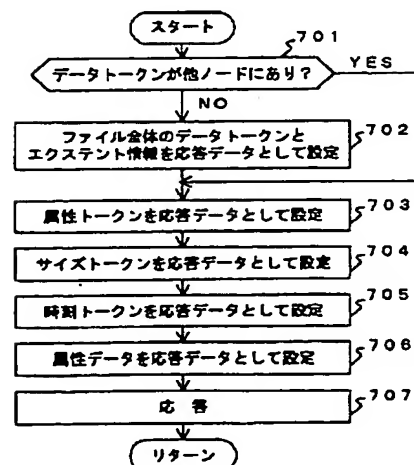
【図2】

ノード内のソフトウェア構成図



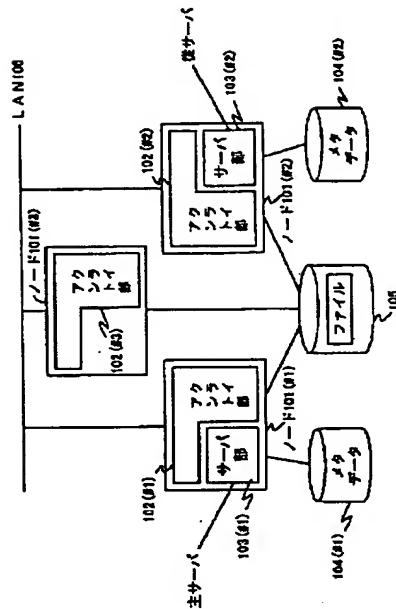
【図7】

サーバ部のopen操作処理の動作フローチャート



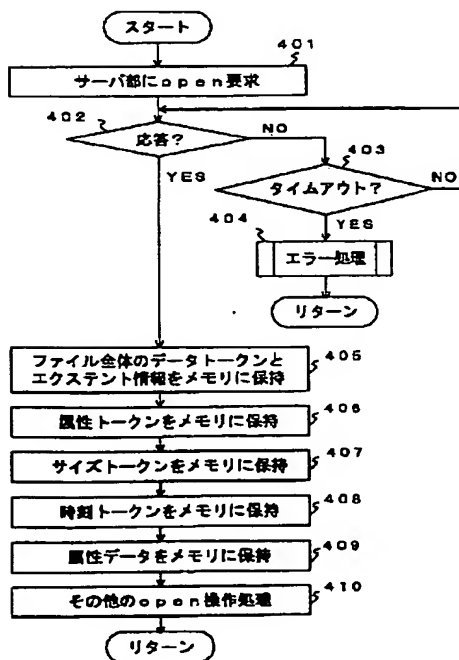
【図1】

本発明の実施の形態のシステム構成図



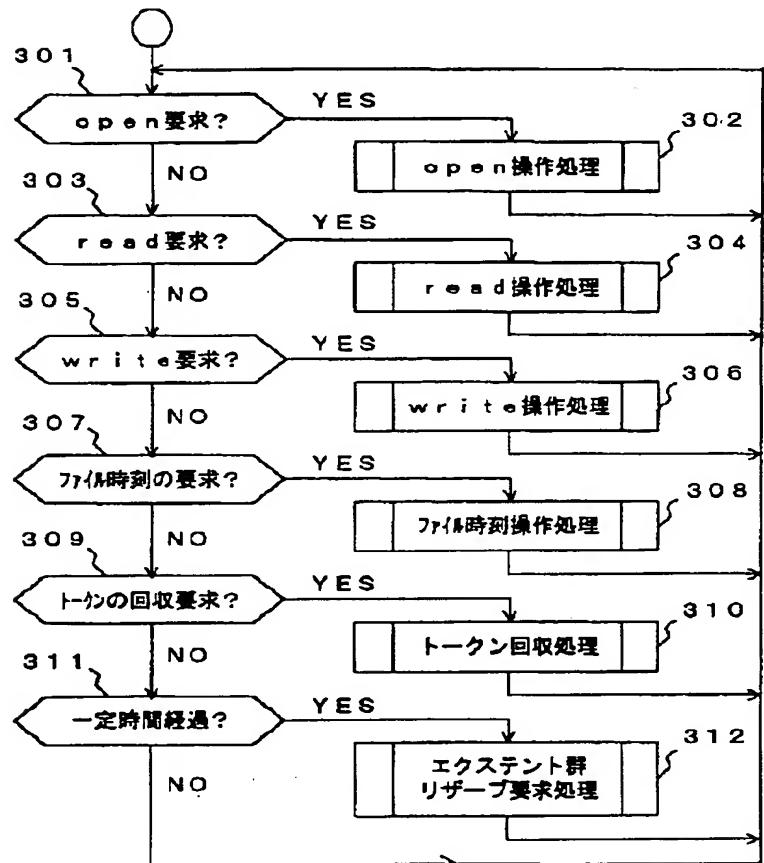
【図4】

クライアント部のopen操作処理の動作フローチャート



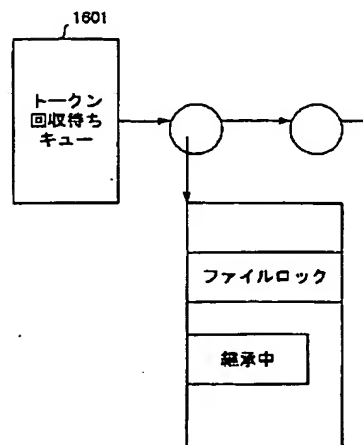
【図3】

クライアント部のメイン動作フローチャート



【図16】

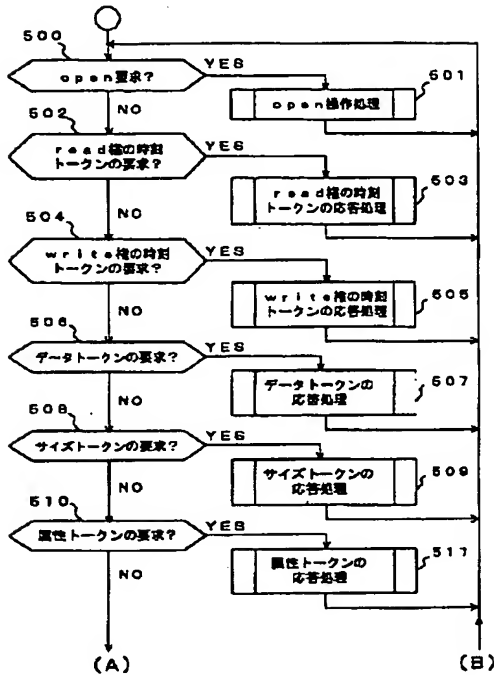
ロック継承制御処理の説明図



トークン回収制御表 1602

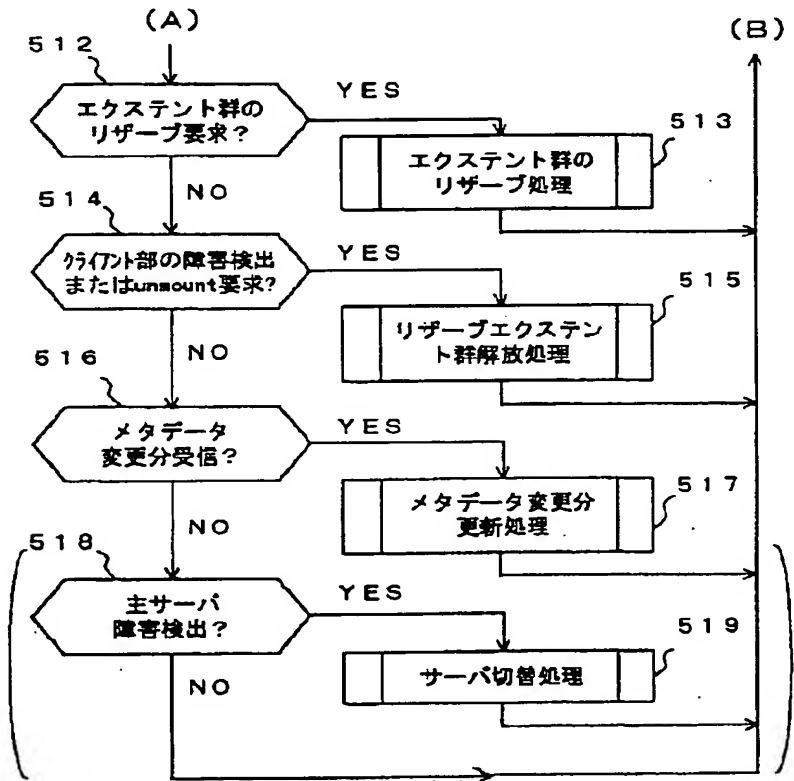
【図5】

サーバ部のメイン動作フローチャート（その1）

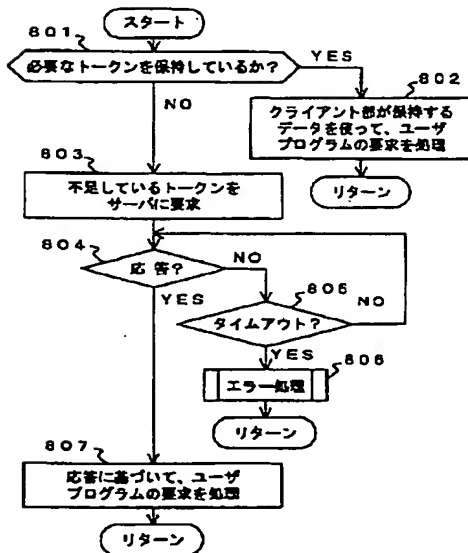


【図6】

サーバ部のメイン動作フローチャート（その2）

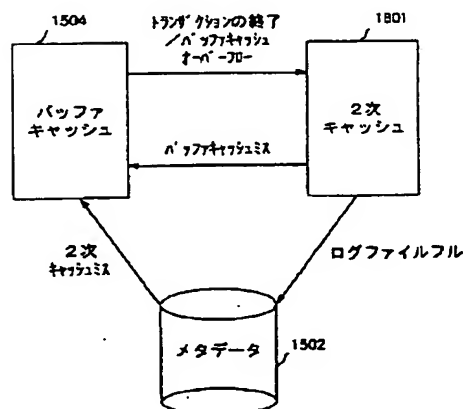


【図8】

クライアント部のread/write操作処理の  
動作フローチャート

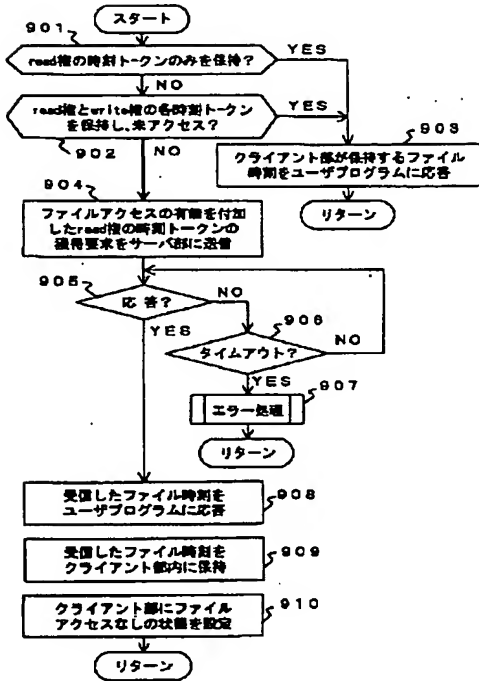
【図18】

ログファイルの2次キャッシュ制御の説明図



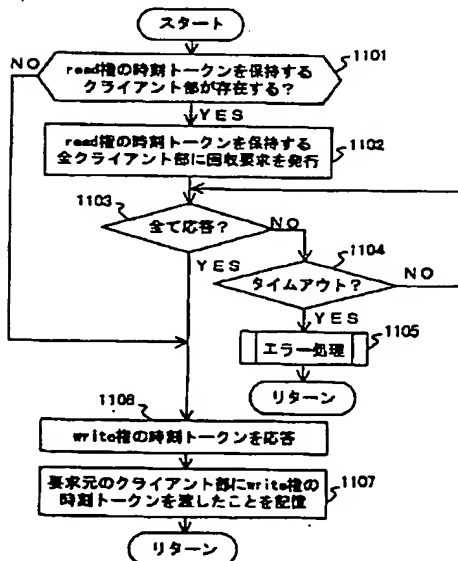
【図 9】

クライアント部のファイル時刻操作処理の  
動作フローチャート



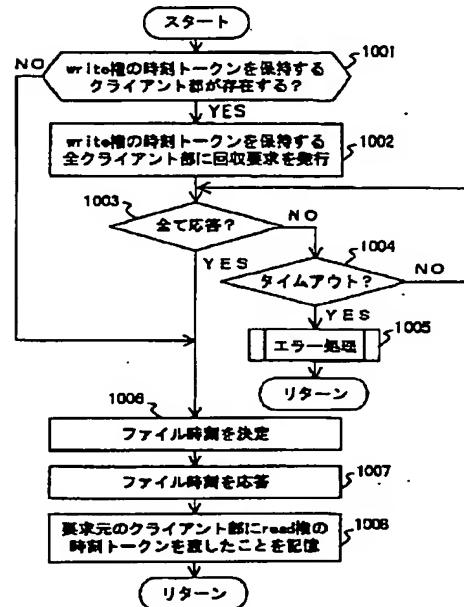
【図 11】

サーバ部でのwrite権の時刻トークンの  
応答処理の動作フローチャート



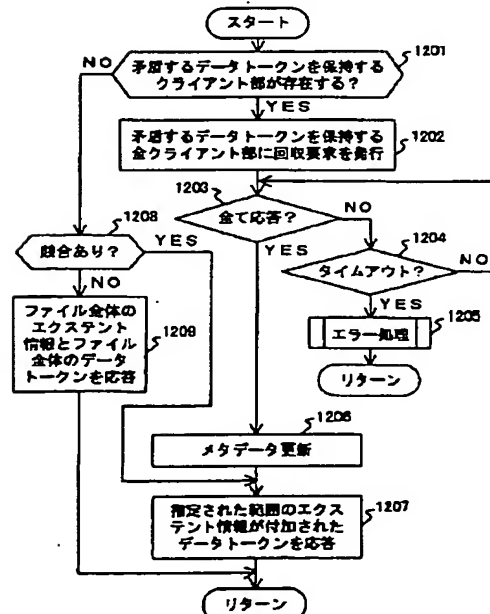
【図 10】

サーバ部でのread権の時刻トークンの  
応答処理の動作フローチャート



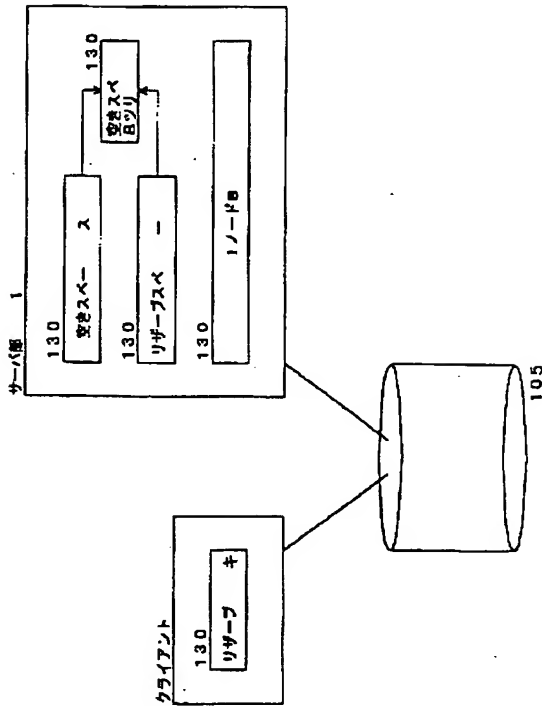
【図 12】

サーバ部でのデータトークンの  
応答処理の動作フローチャート



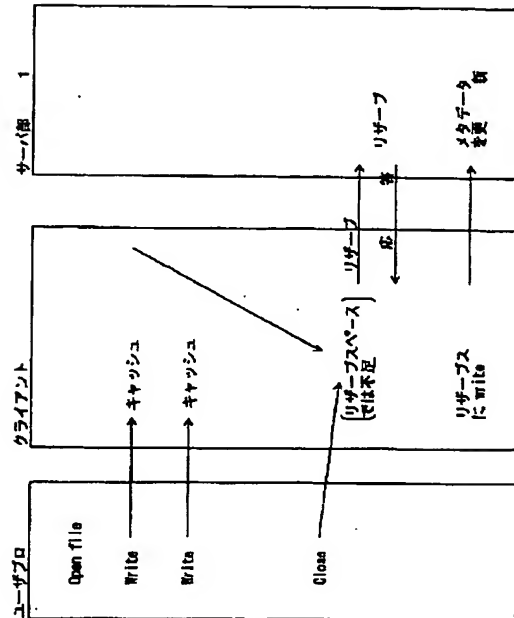
【図13】

エクステンツ管理の詳細を示す図

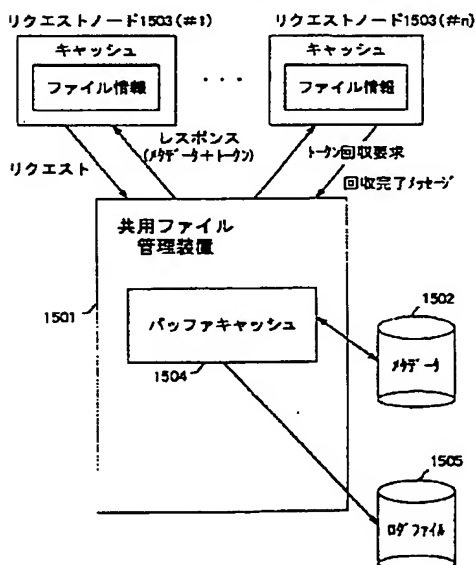


【図14】

エクステンツ管理のシーケンス図

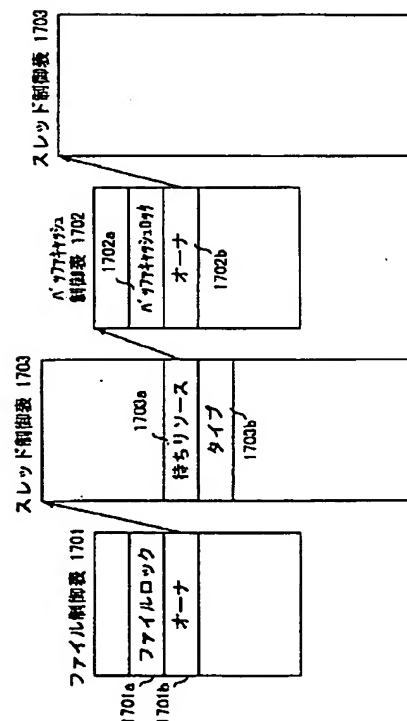


【図15】

ログ制御機構を安装了  
ノード間ファイル共用管理システムの基本構成図

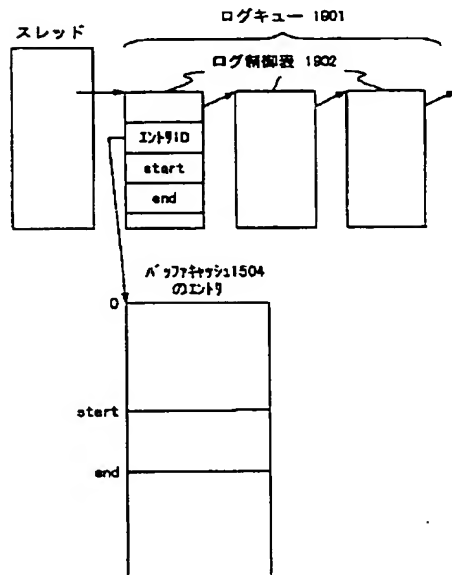
【図17】

デッドロック検出処理の説明図

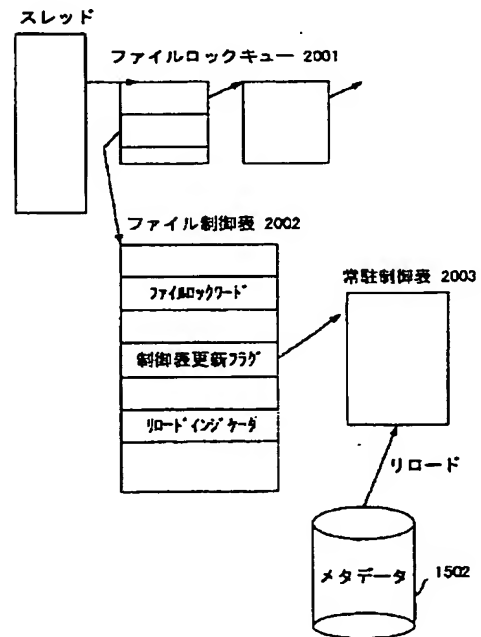


【図19】

ログデータ量を削減できるログ制御処理の説明図



【図20】

トランザクションキャンセル時における  
メモリ常驻制御表のリトスア制御処理の説明図

フロントページの続き

(72) 発明者 村上 岳生  
 神奈川県川崎市中原区上小田中4丁目1番  
 1号 富士通株式会社内

Fターム(参考) 5B082 DE03 EA07 FA18 HA02  
 5B089 GA11 GA21 GB02 HA06 JB15  
 KA06 KB09 KB11



**This Page is Inserted by IFW Indexing and Scanning  
Operations and is not part of the Official Record.**

## **BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☒ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: \_\_\_\_\_

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.**